Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
Санкт-Петербургский государственный университет

Козлов Денис Игоревич

РАЗВИТИЕ МЕТОДОВ ПРОГНОЗИРОВАНИЯ ФИНАНСОВЫХ ВРЕМЕННЫХ РЯДОВ С ПОМОЩЬЮ КОМИТЕТОВ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

Выпускная квалификационная работа по направлению подготовки
035300/50.03.01 «Искусства и гуманитарные науки»

Профиль подготовки «Сложные системы»

Научный руководитель Юрий Александрович Куперин,
доктор физ.-мат. наук, профессор СПБГУ

_____

подпись, дата

Санкт-Петербург
2017

Оглавление

Введение

Фондовый рынок является частью финансового рынка, на котором происходит купля и/или продажа ценных бумаг. Многообразие и характеристики финансовых инструментов, обращающихся на нем, подробно описаны в различных учебных пособиях [1;2]. Стоит отметить, что рынок является динамической структурой и находится в постоянном развитии, тем самым, регулярно предоставляя специалистам новый материал для исследования.

Под ценной бумагой понимается документ, удостоверяющий, с соблюдением установленной формы и обязательных реквизитов, имущественные права, осуществление или передача которых возможны только при его предъявлении [1].К ценным бумагам относят, например, облигации, депозитарные расписки, чеки, коносаменты, закладные, векселя и акции. Исследованию динамики цен акций и их прогнозированию будет посвящена основная часть данной работы.

Акция – эмиссионная ценная бумага, закрепляющая права ее владельца (акционера) на получение части прибыли акционерного общества в виде дивидендов, на участие в управлении акционерным обществом и часть имущества, остающегося после его ликвидации[1]. Этот вид ценных бумаг имеет свою классификацию, категории и свойства[2]. Поскольку акция является инструментом инвестирования и прироста капитала, ее инвестиционному потенциалу может быть дана оценка, которая, в свою очередь, зависит от ряда характеристик, основными из которых являются надежность эмитента, ликвидность и доходность. Акции или другие ценные бумаги эмитентов, лидирующих по этим показателям, принято называть голубыми фишками. Официального определения этого термина в экономической науке не существует, но, как правило, под ними понимаются финансовые инструменты наиболее капитализированных, надежных и

ликвидных компаний, показывающих стабильные показатели выплачиваемых дивидендов. Можно сказать, что голубые фишки являются индикаторами всего рынка, поскольку их котировки лежат в основе расчета основных фондовых индексов мира. Не существует общепринятых объективных критериев, по которым ценную бумагу компанию относят к разряду «голубых фишек» или «второму эшелону», но можно точно сказать, что первые характеризуются стабильностью, лидирующими показателямиежедневных оборотов, а компании-эмитенты – стабильными показателями доходности и прозрачной финансовой деятельностью. Примерами таких компаний на отечественном рынке, согласно данным московской биржи [3] являются АО «Газпром», ПАО «Татнефть», ПАО «Сбербанк», ПАО «Лукойл», ПАО «Ростелеком» и т.д. На американском фондовом рынке – корпорации «AT&T», «Merck&Co», «InternationalBusinessMachinesCorp», «Microsoft», «McDonald's», «JPMorgan&Chase»и т.д.

Известно [4], что подавляющее большинство сделок на биржах являются спекулятивными, т.е. направленные на получение прибыли по схеме «купить дешевле, продать дороже». Такие сделки совершаются на основе предсказаний рыночными агентами движений финансовых инструментов, то есть цен тех или иных технических индикаторов.Точность прогноза зависит от эффективности экономико-математических методов, применяемых при обработке, анализе и предсказании временных рядов котировок.

Прогнозирование финансовых временных рядов – одна из ключевых задач инвестиционной аналитики. При всем многообразии количественных методов анализана текущий момент не существует общепринятой модели прогнозирования, позволяющей получить одинаково качественные предсказания для различных финансовых инструментов на разные временные промежутки. Существующая методология не позволяет осуществлять абсолютно достоверные предсказания, что приводит к

необходимости дополнительных исследований с целью улучшить уже известные методы. В этом и состоит актуальность данной дипломной работы.

Целью данной работы является изучение существующей методологии в области предсказания финансовых временных рядов с помощью искусственных нейронных сетей, а также разработка собственного метода прогнозирования на основе комитетов искусственных нейронных сетей для эффективной торговли финансовыми инструментами.

Для достижения поставленной цели в настоящей дипломной работе решены следующие задачи:

1. Получение исходных данных котировок цен финансовых временных рядов с необходимой частотой дискретизации;
2. Предобработка и фильтрация полученных данных;
3. Выбор архитектуры и метода обучения нейронной сети и формирование входного пространства признаков для получения прогнозов;
4. Формирование комитета нейронных сетей и апробирование различных подходов получение прогноза комитетом;
5. Анализ полученных результатов.

Естественнонаучный подход, а также методы, заимствованные из таких областей физики и прикладной математики, как нелинейная динамика и нейроинформатика дают возможность сочетать нетипичные подходы для решения задачи прогнозирования для достижения эффективного результата. Обзор использованных методов нелинейной динамики приведен в теоретической части данной дипломной работы.

Все компьютерные вычисления были реализованы в пакете MATLAB, вся работа с нейронными сетями была проведена в тулбоксе

«NeuralNetworkToolbox» пакета MATLAB. Исходные ряды котировок цен акций были скачаны с официального сайта компании ФИНАМ [5].

Часть I. Теория

Фильтрация временного ряда.

Механизм ценообразования финансового инструмента является сложным процессом. Под характеристикой «сложный» понимается влияние многих факторов на динамику цен. Изменение цены может являться прямым следствием финансовой деятельности компании, сезонной составляющей, действий крупных инвесторов и спекулянтов, косвенным следствием политических событий, распространения ложной информации, количество факторов не поддается исчислению. Исследуя временной ряд котировок стоит понимать, что на его динамику оказывают влияние компоненты, выделение и отдельный анализ которых может быть затруднен или не возможен, а их взаимодействие зачастую нивелирует или искажает ключевые закономерности, которые хочет обнаружить исследователь. В связи с этим появляется необходимость в предобработке.

Рассматривая временной ряд с точки зрения методов анализа нелинейной динамики, можно сказать, что график котировок является индикатором состояния рыночной системы, при этом количество степеней свободы конечно и равно числу рыночных агентов [7]. В этом контексте приобретает значение уровень влияния (сильное или слабое) отдельного агента на динамику ряда. Крупные игроки в данном случае будут оказывать более весомое воздействие, а действия мелких спекулянтов можно расценить как случайную составляющую – шумовую компоненту. Чем выше уровень шума, тем более вклад случайной составляющей в финансовый временной ряд, и, как следствие, тем хуже качество прогноза нейронной сети [7]. Таким образом, для повышения точности предсказания необходимо отфильтровать

ряд путем выделения и отбрасывания шумовой компоненты. Благодаря этому, нейронной сети будет проще выделять закономерности, скрытые в обучающей выборке. Для решения этой задачи в данной дипломной работе применяется метод Empirical Mode Decomposition[6].

Empirical Mode Decomposition (EMD) – метод разложения сигнала на функции, которые получили названия «эмпирические моды»[6].

Метод EMD представляет собой итерационную вычислительную процедуру, в результате которой исходный сигнал, непрерывный или дискретный, раскладывается на внутренние колебания (IMF–Intrinsic Mode Functions [6]). Важно отметить, что каждая из мод не задается отдельными формулами, а определяется исходным временным рядом. Иными словами, передаточные функции преобразования определяются самой последовательностью, а не линейными подсистемами с постоянными параметрами. Благодаря этому свойству, метод EMD принадлежит к классу адаптивных фильтров.

Результатом разложения исходного сигнала является группа IMF функций, упорядоченных по частоте от наибольшей к наименьшей. Так, первая мода визуализирует наиболее зашумленную компоненту ряда, а последняя –монотонный остаток или тренд.

Алгоритм EMD-фильтрации разлагает исходный сигнала $X(t)$ на функции эмпирических мод $c_j(t)$ и остатков $m(t)$.Результатом декомпозиции будет представление сигнала в виде суммы модовых функции и конечного остатка:

$$X(t) = \sum_{j=1}^{n} c_j(t) + m(t),$$

где t - количество эмпирических мод, которое устанавливается в процессе вычислений.[6]

Порядок вычисления моды следующий[6]:

1. В исходном сигнале $\mathrm{X}(t)$ вычисляются все экстремумы;
2. Нижние точки, минимумы, соединяются между собой. Эта процедура называется интерполяцией. Получившаяся линия называется нижней огибающей, $e_{\min}(t)$. Аналогично получается верхняя огибающая $e_{\max}(t)$;
3. По формуле $m(t) = (e_{\max}(t) + e_{\min}(t))/2$ рассчитывают среднюю величину;
4. Вычисляем первое приближение к первой функции моды. $h(t) = x(t) - m(t)$
5. Повторяем предыдущие операции, только вместо исходного сигнала $\mathrm{X}(t)$, теперь используем первое приближение $h(t)$. Эта итеративная процедура просеивания (sifting) повторяется до тех пор, пока среднее не обратится в ноль. Таким образом, вычисляется первая мода $c_1(\mathrm{t})$.
6. Из исходного сигнала вычитаем первую моду, вычисляем остаток. Производим все предыдущие операции с остатком. $\mathrm{X}(t) - c_1(\mathrm{t}) = m(\mathrm{t})$

Процесс декомпозиции заканчивается, когда остаток становится монотонной функцией без экстремумов, из которой уже нельзя будет выделить эмпирическую моду[6].

О реконструкции аттракторов

Задача обучения отдельной сети на первом этапе и комитета – на втором, сводится к формированию эффективной структуры входных данных. Под структурой входных данных понимается отображение исходного ряда в набор обучающих примеров, при этом эффективность этой структуры оценивается по ошибке прогнозирования [9]. В данной дипломной работе алгоритмом отображения исходного ряда в набор обучающих примеров является погружение ряда в лаговое пространство [9]. В свою очередь, алгоритм погружения является промежуточной задачей при реконструкции аттрактора динамической системы.

Идея реконструкции предшествовала решению задачи нейросетевого прогноза. Первые достижения в этом направлении связаны с исследованиями в области сложного и нерегулярного движения потоков, называемого турбулентностью[11].В этом контексте имеет смысл упомянуть о статье Ф. Такенса и Д. Рюэля «О природе турбулентности», новаторское значение которой заключается в соединении физической модели с математическими идеями теории динамических систем, что дало мощный толчок к развитию последних.

Появление странных аттракторов [13] существенно расширило возможности построения математических моделей. Для динамических систем в непрерывном времени в фазовом пространстве как правило существует аттрактор, к которому «притягиваются» траектории ее состояний при стремлении времени к бесконечности. Аттрактор системы может предоставить исследователю информацию о параметрах динамической системы, а также, может быть использован для прогнозирования ее состояний в будущем. Если у исследователя достаточно информации о системе, он может построить в фазовом пространстве ее аттрактор и, тем самым, реконструировать математическую модель [15].

Наличие математической модели позволяет получать данные о параметрах системы, и, как следствие, позволяет прогнозировать динамику ее развития в будущем. На практике же исследователь может столкнуться с недостаточностью информации о системе или ее отсутствием. В таком случае, остается только наблюдать за системой и регистрировать ее сигналы. В случае, если доступно наблюдение хотя бы за одной динамической переменной, то эту переменную можно идентифицировать, как *наблюдаемую*[16].

Примером наблюдаемой в рыночной системе является временной ряд котировок цен какого либо финансового инструмента. В соответствии с современными парадигмами теории динамических систем [15], можно утверждать, что по одномерной реализации можно реконструировать фазовое пространство и в этом реконструированном пространстве восстановить так называемый псевдоаттрактор или реконструированный аттрактор системы. Это утверждение основывается на теореме Такенса, подробное описание которой приведено ниже. Перед этим необходимо определить терминологию, на которой она основывается.

Отображения. Многообразия. Погружения.

В этом разделе приводятся математические сведения, необходимые для понимания теоремы Такенса. Метрическое пространство (M, d) – это множество M вместе с вещественной функцией d, которая удовлетворяет аксиомам метрики. Соответствие между различными множествами будут описаны функциями или отображениями: $f(*)$ или $f: X \to Y$. Подмножество $\{f(x)\} \in Y$ - это область значений функции, а $\{x\} \in X$ – ее область определения. Для $\mathrm{M} \equiv R^n$, где R^n - n-мерное евклидово пространство (обозначается как $n = \dim M$) [8].

Пусть (M, d) – метрическое пространство и $r > 0$ – вещественное число. Тогда *открытым* шаром в точке $a \in \mathrm{M}$ называется подмножество $B(a,r) \subset \mathrm{M} : B(a,r) = \{x \in \mathrm{M} \mid d(a,x) < r\}$. Шар называется *замкнутым*, если $d(a,x) \le r$.

Отображение $f : A \to B$ называют *сюръекцией*[8] или *отображением на*, если образ всего A совпадает с B, т.е. каждый элемент из B является образом по крайней мере одного элемента из A. Если $A = B$, то элемент x, удовлетворяющий условию $x = f(x)$ $(f : A \to A)$ называют *неподвижной точкой* отображения f. Отображение $\varphi : A \to B$ называют *инъективным*[8] или *взаимно однозначным*, если $\forall y \in B$ существует не более одного элемента $x \in A$ такого, что $y = \varphi(x)$. Тогда $\varphi(x) = \varphi(y)$ влечет $x = y$. Если $B = R^n$, то f называют функцией. Обратная к $f(x)$ функция будет обозначаться $f^{-1}(x)$. Отображение, являющееся инъективным и сюръективным, называется *биекцией* [8].

Отображение $f : X \to Y$ *непрерывно* в $x \in X$, если для любой окрестности V, открытой в Y и содержащей точку $f(x)$, найдется такая открытая в X окрестность $U(x)$, что $f(U) \subset V$. Отображение φ открытого множества U из R^n в R^m называется *гладким* [8]. Если оно имеет непрерывные частные производные вплоть до некоторого r говорят, что φ принадлежит классу C^r.

Рассмотрим два множества A и B таких, что $A \subset B$. Говорят, что A *плотно*[8] в B, если:

- каждая точка B либо принадлежит A, либо является точкой накопления A, (т.е. каждая точка x, принадлежавшая или не

принадлежавшая A, содержит в своей окрестности по меньшей мере одну точку $y \neq x \mid y \in B$), или

- каждое открытое подмножество точек B содержит точку A.

Многообразия

Рассмотрим группу преобразований плоскости R^2

$$x' = ax + by + c,\ y' = px + qy + s,\ \| A \| = \begin{pmatrix} a & b \\ p & q \end{pmatrix},$$

заданную матрицей $\| A \|$. Если $\| A \|$ - ортогональная, то обычную геометрию можно определить, как совокупность тех свойств геометрических объектов, которые сохраняются (инвариантны) относительно указанной выше группы преобразований. Например, заведомо сохраняются расстояния между точками и углы между прямыми. *Аффинная геометрия* получается [8], если ортогональность матрицы $\| A \|$ заменить на требование $\det A \neq 0$. Следующим обобщением будет группа, заданная в виде:

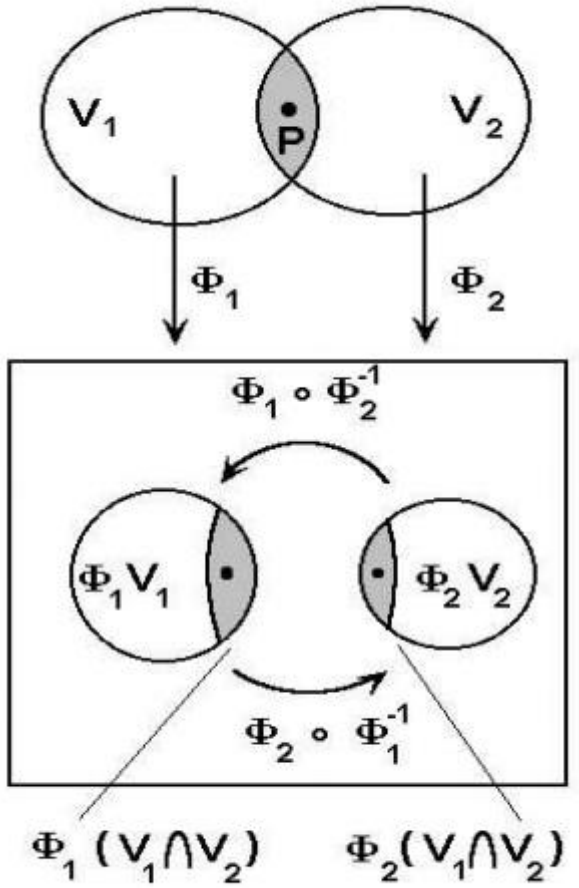
$$x' = f(x, y), y' = g(x, y)$$

где f и g - нелинейные, но C^r-гладкие функции. Если же такими свойствами обладают обратные преобразования $(f^{-1}, g^{-1})^7$, мы получим *геометрию гладких многообразий* [8]. Это совокупность свойств, инвариантных относительно растяжений и деформаций, т.е. *гомеоморфизмов* [8]. В этой геометрии, например, куб, шар и конус неразличимы.

Рассмотрим сферу S^2 и бесконечное число ее копий, не различимых с точностью до гомеоморфизмов. Весь этот класс эквивалентных объектов называют *многообразием* [8]. Привычная сфера лишь типичный

представитель этого класса, его конкретная реализация в евклидовом пространстве R^3.

Компактное многообразие (топологическое многообразие) – топологическое пространство, каждая точка которого обладает окрестностью, гомеоморфной множеству в евклидовом пространстве R^n. Иными словами, пространство, локально сходное с евклидовым. Здесь n – размерность топологического многообразия.



Окрестность точки x многообразия можно определить, указав выделенный класс подмножеств (их называют *открытыми*), все точки которых «достаточно близки к любой точке». Класс замкнут относительно операций объединения и пересечения. А именно, пересечение конечного числа открытых множеств и объединение любого их числа также принадлежит указанному классу. Именно он определяет топологию многообразия [8].

Пусть M – многообразие, U - подмножество в M, а Φ – биекция U в R^n. Тогда n-мерной картой «c» на M называют пару (U, Φ). Здесь U – область карты, а – Φ - координатное отображение. Иными словами, мы просто приписываем каждой точке $p \in U$ n чисел:

$$(\Phi_1(p), \Phi_2(p), ..., \Phi_n(p)) = (x_1, x_2, ..., x_n)$$

которые называются координатами p в карте $c = (U, \Phi)$ Отображение Φ^{-1} называют *параметризацией* U .

Пусть точку p содержат две пересекающиеся окрестности V_1 и V_2 (рисунок 1). Имеем две карты: $c = (V_1, \Phi_1)$ и $c' = (V_2, \Phi_2)$. Эти карты C^r – согласованы, если: $\Phi_1(V_1 \bigcap V_2)$ и $\Phi_2(V_1 \bigcap V_2)$ открыты в R^n и функции

$$\Phi_1 \circ \Phi_2^{-1} : \Phi_2(V_1 \bigcap V_2) \to \Phi_1(V_1 \bigcap V_2)$$

$$\Phi_2 \circ \Phi_1^{-1} : \Phi_1(V_1 \bigcap V_2) \to \Phi_2(V_1 \bigcap V_2)$$

Рисунок 1. Согласование карт на многообразии.[8]

являются C^r -гладкими функциями, т.е. C^r -диффеоморфизмами. Полный набор согласованных карт для всего многообразия называют C^r -*атласом*, а само многообразие C^r - дифференцируемым.

Отображение $\varphi : X \to Y$ двух гладких многообразий или двух произвольных множеств в R^n называют *диффеоморфизмом* [8], если φ и φ^{-1} - гладкие гомеоморфизмы. Окружность в R^2 диффеоморфна эллипсу, но не треугольнику, которому она лишь гомеоморфна. Диффеоморфизмы определяют класс эквивалентных диффеоморфных пространств.

Погружения и вложения

Пусть $f : X \to Y$ гладкое отображение двух многообразий и $\dim X < \dim Y$. Если для этого отображения у каждой точки в X существует окрестность U , которую f гомеоморфно отображает в $f(U)$, то оно называют *иммерсией* или *погружением* X в Y [8].

Каноническая иммерсия [8] получается, если евклидово пространство R^k погрузить в $R^m (k < m)$. При этом точка $x \in R^k$ *линейно* отображается в точку $f(x) \in R^m$:

$$(x_1, x_2, ..., x_k) \to (x_1, x_2, ..., x_k, \underbrace{0, ..., 0}_{m-k}).$$

В этом случае образ иммерсии – подпространство в R^m.

В общем случае гладкое отображение F компактного гладкого многообразия A называют иммерсией, если дифференциал отображения $dF(x)$ является взаимно-однозначным в каждой точке A.

Дифференцируемое *вложение* [8] A - это гладкий диффеоморфизм из A на собственный образ F(A), т.е. гладкое взаимно-однозначное отображение, обратное для которого $F^{-1}(A)$ также гладко. Если A - компактное многообразие, то отображение F является вложением тогда и только тогда, когда F -взаимно-однозначная иммерсия. Для *топологического вложения* диффеоморфизм заменяется на гомеоморфизм.

Теорема Такенса

Пусть D^r – множество C^r - диффеоморфизмов $\{f^t\}$, $f^t : M \to M$, компактного d –мерного многообразия M и пусть $C^r(M, R)$ - множество наблюдаемых функций $h(t), h : M \to R$. Определим для $m \geq 2d+1$ отображения запаздывающих координат $F_{f,h} : M \to R^m$:

$$F_{f,h}(\vec{x}) = (h(\vec{x}), h(f(\vec{x})), ..., h(f^{m-1}(\vec{x}))).$$

Тогда, множество (f^t, h), для которого $F_{f,h}$ является вложением, открыто и всюду плотно в $D^r(M) \times C^r(M, R)$ для $r \geq 1$. [13].

Здесь отображение F , которое реализует вложение, называют отображением запаздывающих координат. R^m - пространство вложения.

Первая часть теоремы сводится, фактически, к тому, что достаточно гладкие (т.е. класса $C^r(M, R)$) наблюдения, например, временной ряд h(t) продуцируются $C^r(M)$ -гладкой динамической системой, которая имеет компактный d -мерный аттрактор.

Вторая часть теоремы предлагает процедуру, как используя временной ряд h(t) , и построить аналог аттрактора в евклидовом пространстве R^m подходящей размерности. $m = 2d + 1$. (Визуализация – рисунок 2.)

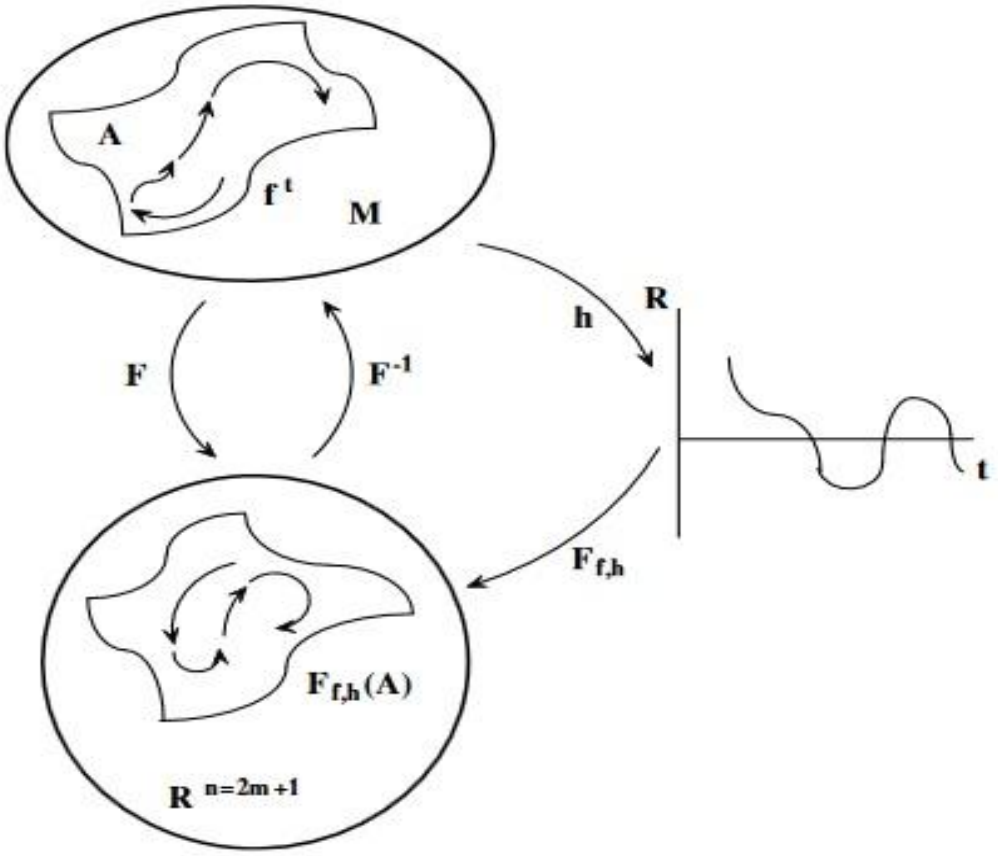


Рисунок 2. Реконструкция аналог аттрактора A из временного ряда h(t) согласно теореме Такенса. [8]

А для этого нужно взять m отсчетов временного ряда, начиная с произвольного номера n.

$$h_n, h_{n+1}, \ldots, h_{n+m-1}$$

Они рассматриваются как компоненты m-мерного вектора и образуют точку образа $z_n \in R^m$. Следующая за ней точка образа получается сдвигом m-мерного набора на один отсчет, а итеративное продолжение этой процедуры – копию истинной орбиты.

Теорема гарантирует [8, 13], что копия будет являться вложением, т.е. сохранит свойства оригинала с точностью до диффеоморфизмов - непрерывных и дифференцируемых преобразований.

Детально разобравшись с механизмом погружения, определим алгоритм формирования входных параметров будущей нейронной сети.

Формирование входных признаков для обучения нейронной сети

Выше было отмечено, что по одномерной реализации можно восстановить фазовый портрет системы. В данной дипломной работе эта процедура решается с помощью искусственных нейронных сетей. Поскольку аттрактор системы воссоздается с помощью лаговых векторов, они и послужат входами для сети. Таким образом, чтобы спрогнозировать динамику системы нужно решить промежуточную задачу – разложить исходный ряд на лаговые вектора и сопоставить каждому из них выход сети. Ниже будет описан процесс формирования обучающего множества.

Пусть изначально наш временной ряд задан следующим образом:

$$X = \{x(i) : 0 \le i \le n\}$$

Лаговые векторы есть отображение исходного ряда в набор обучающих примеров при помощи параметров реконструкции [9]. Такими параметрами являются:

m - размерность пространства вложения;

τ - временная задержка, лаг;

Каждый обучающий пример состоит из входа:

$$X_i = [x(i), x(i-\tau), x(i-2\tau), ..., x(i-m+1)\tau],$$

и выхода нейронной сети:

$$Y_i = x(i+1)$$

Дальнейшие необходимые вычисления на этапе подготовки обучающего множества – расчет параметров реконструкции. Вычисление m и τ - ключевая задача на этапе подготовки к нейросетевому прогнозу. От решения этой задачи напрямую зависит качество обучения нейронной сети: выбор τ влияет на скоррелированность соседних значений вектора, а ошибочный выбор численного значения параметра m может привести к избыточности или недостатку информации при обучении ИНС.

Определение временной задержки

Для определения временной задержки существует несколько методов. Наиболее популярны следующие методы расчета временного лага[10]:

- Метод автокорреляционной функции;
- Метод взаимной информации;
- Метод среднего отклонения;

В данной дипломной работе к разным финансовым инструментам были применены первые два метода.

Автокорреляционная функция (АКФ)- зависимость взаимосвязи между сигналом y_i и его сдвинутой копии $y_{i+\tau}$ от величины временного сдвига.

$$c_\tau = \frac{1}{T-1}\sum_{t=1}^{T-\tau}(\mathrm{y}_t - \bar{\mathrm{y}})(\mathrm{y}_{t+\tau} - \bar{y})$$

Здесь $\bar{y}$ выборочное среднее временного ряда y_i размером T.

Поскольку вычисления проводились в среде MATLAB, формула АКФ взята из библиотеки функций. Для заданного значения τ функция АКФ означает коэффициент корреляции между исходным рядом и копией, взятой с задержкой на τ шагов.

Временная задержка выбиралась численно равной времени первого пересечения нуля функцией.

Метод взаимной информации

Пусть $(a,\mathrm{b}) \in \mathrm{R}$ – минимальный интервал, содержащий все значения исходного ряда. Разобьем его на L равных частей. Количество интервалов определяется формулой $L = ([\log_2 \mathrm{N}] + 1)$.Здесь N - количество отсчетов в исходном ряде [10].

Обозначим событие «значение $x(t)$ принадлежит i-му интервалу» через A_i, а событие «значение $x(t+\tau)$ принадлежит j-му интервалу» через B_i. Тогда *функция взаимной информации* $\mathrm{I}(\tau)$ определяется следующим образом [10]:

$$\mathrm{I}(\tau) = -\sum_{i=1}^{L}\sum_{j=1}^{L} P(\mathrm{A}_i \mathrm{B}_j) \cdot \log_2 \frac{P(\mathrm{A}_i, \mathrm{B}_j)}{P(\mathrm{A}_i)P(\mathrm{B}_j)}$$

Где $P(\cdot)$ - вероятность соответствующего события.

$P(\mathrm{A}_i, \mathrm{B}_j) = P(\mathrm{A}_i \mid \mathrm{B}_j)P(\mathrm{B}_j)$, где $P(\mathrm{A}_i \mid \mathrm{B}_j)$ - условная вероятность события A_i при наступлении события B_j.

При использовании этого метода значения временного лага выбиралась в соответствии с первым минимумом функции $\mathrm{I}(\tau)$.

Выбор размерности вложения

Из теоремы Такенса следует, что размерность пространства вложения равна числу степеней свободы системы или количеству переменных, достаточных для ее описания. Также, по этому параметру можно сделать вывод о сложности системы. Существует несколько способов вычислить параметр m [10]:

- метод корреляционной размерности;
- метод ближайших ложных соседей;
- гамма-тест.

В данной дипломной работе размерность пространства вложения [10] вычислялась методом ближайших ложных соседей.

В основе алгоритма лежит следующее следствие теоремы Такенса: при верном вычислении параметров реконструкции оригинальный и псевдоаттрактор должны быть топологически идентичны. Поскольку фазовые траектории оригинального притягивающего множества не пересекаются, аналогичная ситуация должна происходить и с реконструированным аттрактором. Нарушение этого правила означает, что вычисленный параметр m меньше фрактальной размерности оригинала. Из теоремы Такенса следует [10], что все ближайшие друг к другу точки аттрактора, восстановленного в R^m, будут также топологически близки и в R^{m+1}. Метод ближайших ложных соседей позволяет вычислить наименьшее значение параметра m, при котором переход к размерности $(m+1)$ будет сопровождаться минимальным смещением «соседей». Вычисленное этим методом значение определит минимальную размерность пространства вложения, при котором траектории псевдоаттрактора не будут пересекаться [10]. Процесс вычисления состоит из следующих этапов:

1. Положим m = 1. Вычисляем для каждой точки $\bar{x}(j)$ ближайшего «соседа» $\bar{x}(\mathrm{i})$ в m -мерном пространстве с помощью вычисления расстояния по формуле: $\| \bar{x}(j) - \bar{x}(\mathrm{i}) \|$.

2. Вычислим расстояние на следующем шаге $\| \bar{x}(j+1) - \bar{x}(\mathrm{i}+1) \|$

3. Определяем отношение $\frac{\| \bar{x}(j+1) - \bar{x}(\mathrm{i}+1) \|_m}{\left\| \bar{x}(j) - \bar{x}(\mathrm{i}) \right\|_m} = R_j$

4. Если вычисленное значение превышает заданный порог $R_j > R_t$, то точка $\bar{x}(\mathrm{i})$ будет *ложным соседом* для $\bar{x}(j)$. После проведения всех итераций подсчитывается общее количество ложных соседей P для каждой точки.

5. Считается P / N и весь цикл повторяется заново для $m = m+1$. Алгоритм повторяется до тех пор, пока отношение P / N не приблизится к нулю. Здесь N - общее количество точек восстановленного пространства, а частное P / N в данном случае выступает мерой достоверности текущего значения m , поскольку определяется соотношением количества ложно восстановленных точек к их общему числу.

Пороговое значение R_t в настоящей работе полагалось равным 2.

Нейронные сети

Искусственные нейронные сети(ИНС) -математические модели, а также их программные или аппаратные реализации, построенные по принципу

организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма. Такое определение появилось во время изучения мозговых процессов и первых попыток их моделирования. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. Авторство пионерской модели принадлежит Дж. Маккалоку и У. Питтсу [17] и относится к 40-м годам XXвека. Построение первой модели ИНС привлекло внимание научного сообщества: в 1948 Норберт Винер публикует [18] работу, в которой математическими методами описывает мозговые процессы, а уже через год был предложен [18] первый алгоритм обучения. Впоследствии, после разработки алгоритмов обучения, получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управлениями др.

Структура ИНС это набор искусственных нейронов, соединенных и взаимодействующих между собой с помощью синапсов. Принцип работы отдельного нейрона схож с функционалом процессора: в его задачи входит принятие, обработка и передача сигнала. Чем больше таких микропроцессоров соединено в общую сеть, тем более сложные задачи они способны решать.

С точки зрения машинного обучения, нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации т.п. Сматематической точки зрения, обучение нейронных сетей — это многопараметрическая задача нелинейной оптимизации. С точки зрения кибернетики, нейронная сеть используется в задачах адаптивного управления и как алгоритмы для робототехники. С точки зрения развития вычислительной техники программирования, нейронная сеть — способ решения проблемы эффективного параллелизма [20]. А с точки зрения искусственного интеллекта, ИНС является основным направлением в структурном подходе по изучению возможности

моделирования естественного интеллекта с помощью компьютерных алгоритмов.

Отличительная особенность нейронных сетей – способность к выучиванию закономерностей в данных. Способность к обучению и идентификации нелинейных закономерностей – одно из главных преимуществ ИНС перед традиционными методами.

Технически процесс обучения реализуется путем координации весовых коэффициентов связи между отдельными нейронами.

Искусственный или математический нейрон

Рассмотрим математическую модель нейрона [20, 21]. Типичный нейрон производит простейшую операцию - взвешивает значения своих входов со своими же локально хранимыми весами (также называемыми синапсами) и производит над их суммой нелинейное преобразование (Рис.3).

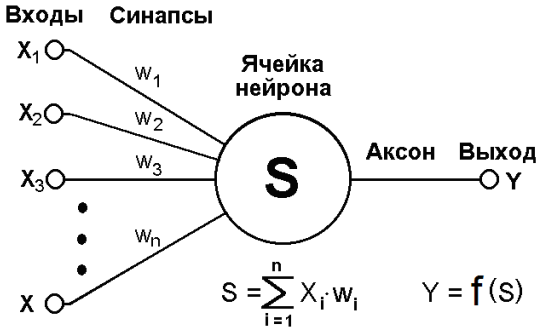


Рис. 3 Математическая модель нейрона [20]

В данных обозначениях:

$X = \{x_1, x_2 ... x_n\}$ - входной вектор,

$W = \{\omega_1, \omega_2 ... \omega_n\}$ - вектор локально хранимых весов (синапсов), $Y = \{y_1, y_2 ... y_m\}$ - вектор выходов,

f - функция активации,

S - внутреннее состояние нейрона.

Функция активации

Функция активации– это функция, вычисляющая выходной сигнал искусственного нейрона[20, 21]. В теле нейрона происходит их суммирование:

$$S = \sum_{i=0}^{n} \omega_i x_i$$

Затем нейрон применяет к сумме некоторую фиксированную функцию f (функцию активации) и выдает на выходе сигнал силы $Y = f(S)$.

Эта модель была предложена МакКалоком и Питтсом еще в 1943 г. [17] При этом использовались ступенчатые передаточные функции (Рис.4), которые определяли формат выходного сигнала Y крайне просто.

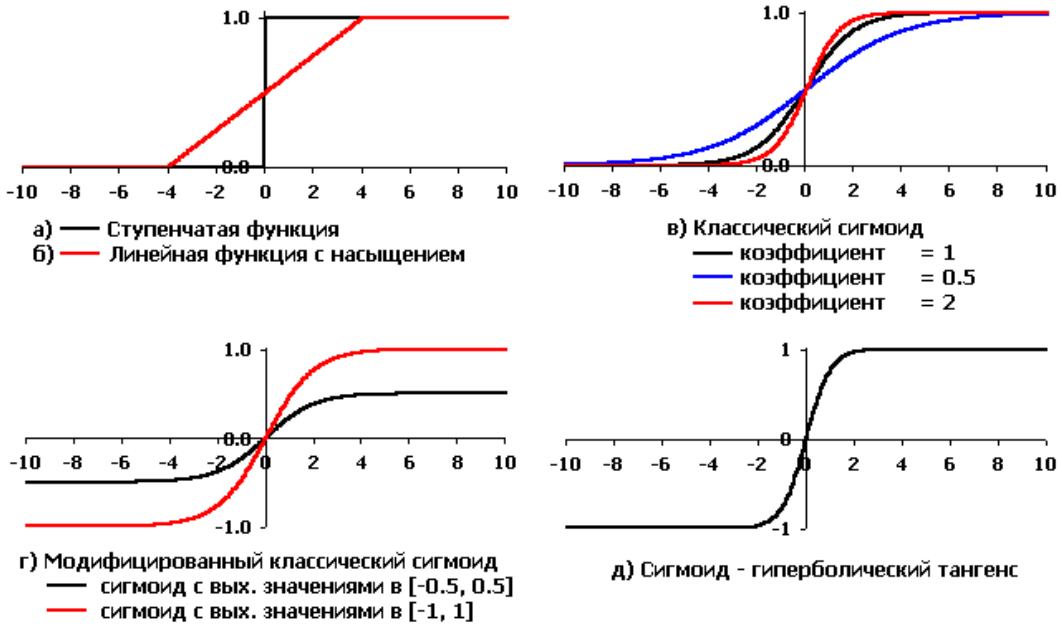


Рис.4 Активационные функции искусственных нейронов, используемые при моделировании ИНС [21]

Функция активации не должна быть линейной функцией. Нелинейность выходной функции активации принципиальна. Известно, что суперпозиция

линейных преобразований есть линейное преобразование. Если бы нейроны были линейными элементами, то любая последовательность нейронов также производила бы линейное преобразование, и вся нейросеть была бы эквивалентна одному нейрону (или одному слою нейронов - в случае нескольких выходов). Нелинейность обуславливает актуальность применения нейросетей при обработке данных о процессах, которые не являются линейными. [20].

Наиболее популярной среди передаточных функций называется логистическая функция. Она задается формулой [19]:

$$f(x)=\frac{1}{1+e^{-\alpha x}}$$

Популярность логистической функции(сигмоиды) обусловлена тем, что в качестве активационной функции она способна усиливать слабые сигналы и ограничивать сильные сигналы.

Архитектура нейронных сетей

Простейшую структуру ИНС можно представить в виде одного «слоя» нейронов, соединенных между собой (рис.5.). Здесь $\overline{X}$ - вектор входов нейросети, $\overline{Y}$ - вектор выходов ИНС, а $\overline{W}$ - вектор синаптических весов.

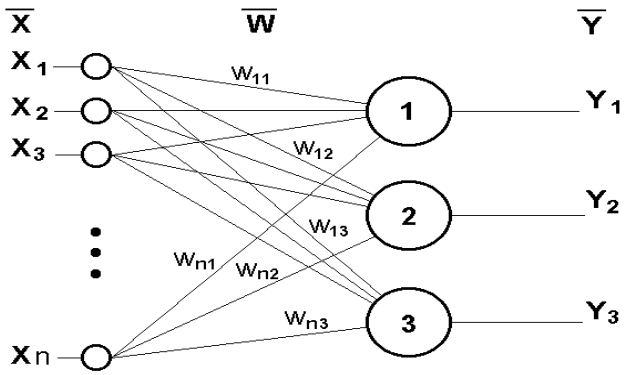


Рис.5 Простейшая нейронная сеть из трех нейронов[27].

.

Одной из первых компьютерных моделей нейросетей стал персептрон. Его математическая модель была предложена Фрэнком Розенблаттом в 1957 году, а ее компьютерная реализация – тремя годами позже [18]. Эта модель сети включает в себя лишь 3 типа элементов:

- сигналы, поступающие от датчиков (нейронов входного слоя);
- ассоциативные элементы (нейроны скрытого слоя);
- реагирующие элементы (нейроны выходного слоя).

Ф.Розенблатт называл такую модель трехслойной, но в контексте современной терминологии она является однослойной. На сегодняшний день однослойный персептрон представляет, в большей степени, исторический интерес, поскольку его обобщающая способность значительно уступает более современным моделям. В то же время, для решения задач аппроксимации многомерных функций широко применяется многослойный персептрон, структура которого изображена на рисунке 6.

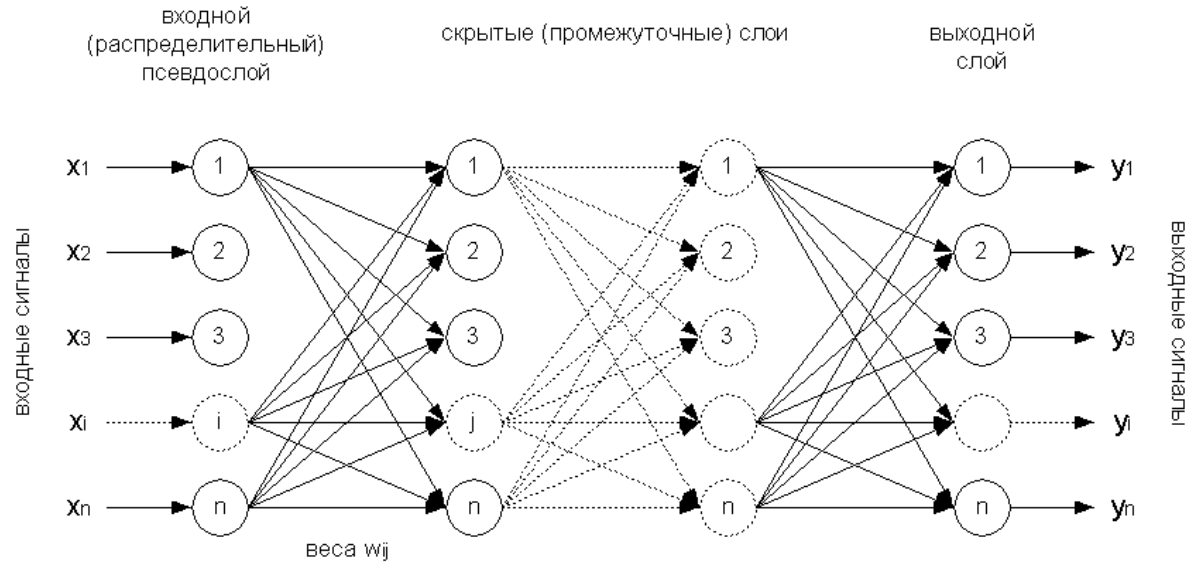


Рис. 6. Структура многослойного персептрона. [22]

В зависимости от типа выходных переменных, аппроксимация функций может принимать вид:

- Классификации (дискретный набор выходных значений)
- Регрессии (непрерывные выходные значения)

Многие более сложные задачи, например, фильтрация шумов или прогнозирование временных рядов, сводится к этим базовым постановкам.

Следует отметить, что при решении задачи прогнозирования у персептрона есть серьезные преимущества перед классическими статистическими методами: авторегрессией проинтегрированного скользящего среднего или экспоненциальным сглаживанием. Это связано с тем, что с математической точки зрения обучение ИНС – это аппроксимация многомерных отображений. Математически, персептрон и является инструментом для решения уравнений с большим количеством неизвестных, поскольку алгоритм поиска последних технически более быстрый [20].

В данной дипломной работе для решения задачи прогнозирования используется именно многослойный персептрон.

Обучение нейросетей

Эффективность применения нейросети для решения типовых задач можно оценить по ошибке, которую эта сеть совершает. Из понимания структуры ИНС следует вывод о том, что ошибка напрямую зависит от синаптических весов. Калибровка весов происходит в процессе обучения, который в общем случае сводится к определению функциональной зависимости $Y = F(X)$, где X и Y являются соответственно входами и выходами нейросети. Оценкой качества определения этой зависимости является функция ошибки [21], которая в общем виде имеет вид: $E(\omega) = E\left\{x^{\alpha}, y^{\alpha}, y(x^{\alpha}, \omega)\right\}$, где $\left\{x^{\alpha}, y^{\alpha}\right\}$ - набор примеров (т.е. пар входов-выходов), на которых обучается нейросеть, а $\left\{y(x^{\alpha}, \omega)\right\}$ - реальные значения выходов нейросети, зависящие от конкретных значений ее синаптических

весов. Способ обучения, когда реальный выход сети сравнивается с эталонным выходом называется обучением с учителем [21].

Цель обучения – определение такого вектора отклонения выхода сети от эталонного ответа, чтобы функционал ошибки принимал минимальное значение. Это превратило бы процесс обучения сети в решение задачи безусловной оптимизации (оптимизация в условиях отсутствия ограничений).

Наиболее популярными методами при решении такой задачи являются:

- алгоритм обратного распространения ошибки;
- алгоритм Левенберга – Марквардта;

Несмотря на широкое применение первого метода, авторы [23, 24] отдают предпочтение алгоритму Левенберга – Марквардта. Главные недостатки алгоритма обратного распространения ошибки – медленная сходимость [24] и негативное влияние локальных минимумов [23]. В данной дипломной работе обучение сети осуществляется методом Левенберга – Марквардта.

Введем следующие обозначения: $X = (x_1, x_2, ..., x_n)$ - входные данные, $Y = (y_1, y_2, ..., y_m)$ - выход сети и параметр w – вектор весовых коэффициентов. На каждом шаге итерации вектор w заменяется на $(w + \Delta w)$ и вычисляется ошибка сети по формуле:

$$F(Y) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{P}(y_{ij} - d_{ij})^2$$

Здесь d_{ij} - желаемый выход j-го выходного нейрона для i-го элемента обучающего множества, y_{ij} - реальные ответы нейросети, а Δw - приращение вектора весов. Упрощенно, суть оптимизации сводится к оценке

приращения. Алгоритм останавливается, если Δw в последующей итерации меньше заданного значения, или если весовые коэффициенты w_n формируют ошибку $F(Y)$ меньше заданного порога. Более подробно с алгоритмом оптимизации можно ознакомиться в [26, 30].

Комитеты и их преимущества

Решая задачу прогнозирования финансовых временных рядов с помощью ИНС, исследователь может столкнуться со следующей проблемой. Сеть определенной архитектуры, качественно обученная по одной выборке и демонстрирующая на ней точные прогнозы может показать совершенно неудовлетворительные результаты при переходе на другой финансовый инструмент или иную частоту дискретизации. Это обуславливается многообразием скрытых паттернов, которые не всегда может выучить отдельная сеть. Встает вопрос о повышении точности прогноза путем обучения сети на разных данных, частотных нарезках и финансовых инструментов. По этому вопросу Терехов пишет [22], что в анализе финансовых данных масштабирование алгоритмов распознавания классов становится одной из основных проблем. Можно выделить 2 способа ее решения: первый заключается в создании и обучении глобального классификатора, второй - в разделении масштабной задачи на ряд более простых и последующем объединении механизмов их решения. Терехов утверждает [22], что вследствие сегментации проблемы может быть повышена точность итогового результата. Основания для такого предположения он связывает с центральной предельной теоремой теории вероятности. Ряд исследователей считают [22,25], что создание и обучение комитета ИНС действительно улучшает точность прогноза.

Получение прогноза комитета искусственных нейронных сетей, как основная задача настоящей дипломной работы, достигается в несколько этапов. Первый заключается в подготовке нескольких десятков отдельных

сетей, обученных на разных выборках и разными алгоритмами. Получив предсказания с помощью каждой сети встает вопрос о том, какие сети подлежат отбору в комитет. Эту процедуру можно провести следующим образом: имея показатели среднеквадратичной ошибки (MSE) и коэффициента детерминации R^2, исследователь может выставить по одному из них/обоим некоторое пороговое значение и сформировать комитет из сетей, прошедших отбор.

Следующий этап, подлежащий формализации, заключается в выборе механизма формирования итогового решения комитета. Ниже рассмотрены наиболее популярные способы организации комитета.

1. Первый и наиболее легкий способ формирования итогового решения комитета заключается в вычислении среднего арифметического прогнозов всех сетей, включенных в комитет.

2. Второй способ заключается в создании «главной сети», которая будет принимать на вход прогнозы отдельных сетей и возвращать прогноз комитета (рис.7). На практике такой подход сталкивается с проблемой недостатка данных для обучения.

Рис.7 Архитектура модели прогнозирования комитета нейроэкспертов [25]

3. Следующий подход заключается в создании «специализации» отдельных сетей. Для реализации такой методики исследователь должен произвести предварительный анализ обучающих выборок с целью формирования групп по принципу схожих паттернов. Так, для финансовых временных рядов, могут быть выделены группы с возрастающим или убывающим трендами. Для каждой группы выбирается несколько сетей, которые будут обучаться только на этой выборке и «специализироваться»на решении конкретной задачи. Также необходимо создать «классифицирующую» сеть, которая будет идентифицировать тип задачи. Предлагается [25] выбрать самоорганизующуюся сеть Кохонена для создания такой сети. и т.п. Для такой кластеризации может быть использована самоорганизующаяся нейронная сеть, называемая картой Кохонена [21].. Стоит отметить, что реализация такого подхода могла бы найти широкое применение не только в контексте решения задачи прогнозирования.

4. Четвертый подход можно обозначить как симбиоз первого и третьего подхода. В этом случае отдельные сети будут обучаться на одной выборке, но каждая будет обладать уникальной архитектурой. Каждая из них по-своему будет решать задачу аппроксимации, благодаря чему возможны более существенные различия в прогнозах. После обучения выборка кластеризуется также, как и в предыдущем примере, после чего каждой сети будет присваиваться коэффициент компетентности для каждого кластера. Решением комитета в данном случае будет весовая функция, учитывающая прогноз каждой, но с различным уровнем значимости.

5. Наконец, в качестве решения комитета можно использовать прогноз лучшей сети. Критерием отбора лучшей сети можно выбрать показатель MSE.

По мнению некоторых экспертов [22, 25], ссылающихся на численные эксперименты [25], применение комитетов нейросетей способно улучшить качество прогноза на 20-30%. по основным статистическим показателям.

При этом, наилучшие результаты показывают сети, для которых были выбраны 4-й и 5-й из описанных выше методов [25].

В настоящей дипломной работе будет исследована гипотеза о повышении точности прогноза в следствие применения комитета ИНС. По результатам численных экспериментов эта гипотеза была подтверждена.

Практическая часть

Цель данной дипломной работы- нейропрогнозирование финансовых временных рядов и оценка результатов прогноза - была достигнута за несколько этапов:

1. Получения исходных данных и их предобработка;
2. Формирование обучающей выборки из обработанных данных;
3. Обучение нейросети и получение прогнозных данных для каждого финансового инструмента;
4. Формирование комитета искусственных нейронных сетей и получение прогноза;
5. Анализ полученных результатов.

Первый этап. Получения исходных данных и их предобработка.

Исходные данные исследования настоящей дипломной работы - временные ряды котировок цен акций некоторых голубых фишек отечественного и американского фондовых рынков. Из российских компаний были выбраны: ПАО «Газпром», ПАО «Аэрофлот», ПАО «Яндекс», ПАО «Сбербанк», ПАО «Мегафон» из американских: «Apple», «McDonald’s», «JPMorgan&Chase». Данные котировок каждого финансового инструмента были скачаны с официального сайта компании Финам с частотой дискретизации 1 день за период пять лет. Таким образом, каждый временной ряд содержит около 1700 отсчетов цен закрытия, «Close», торгового дня. В

качестве примера, графики котировок акций компании «Сбербанк» (рис 7.) и «JPMorganChase» (рис. 8) представлены ниже.

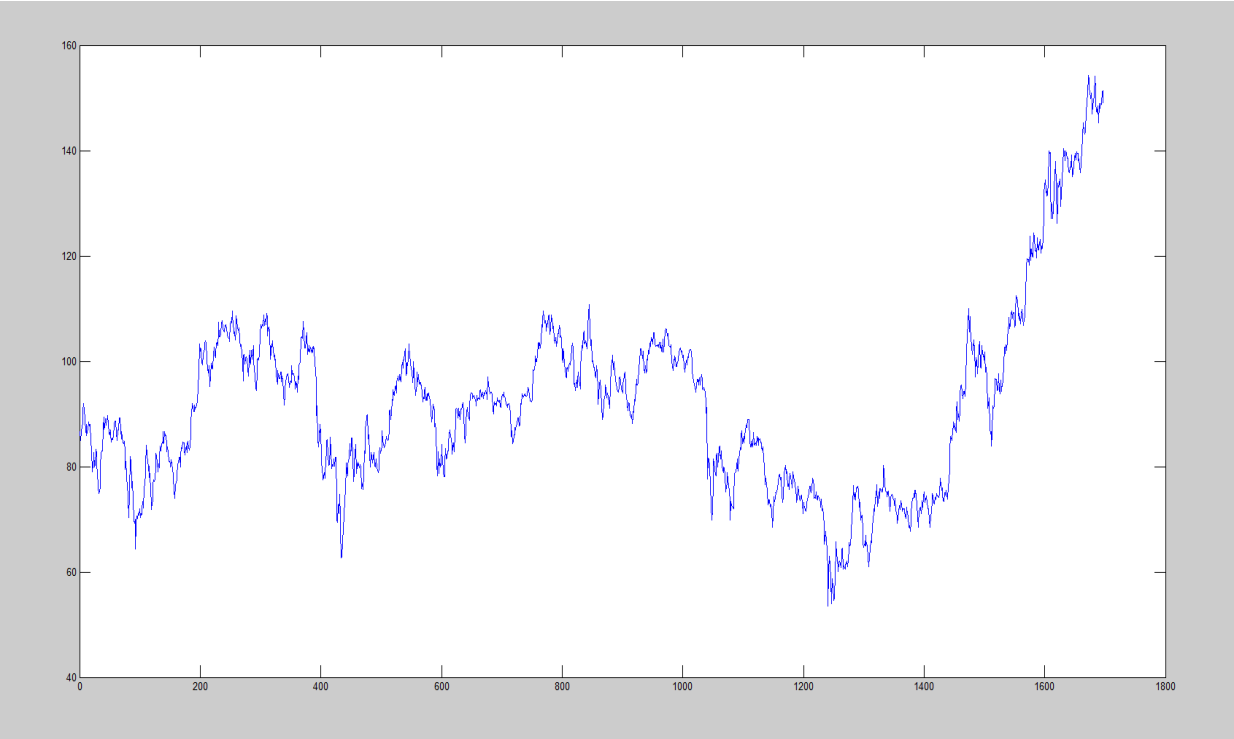


Рисунок 7. Котировки акций компании СБЕРБАНК с частотой дискретизации в 1 день. На графике по оси x – дневные отсчеты, по оси y – цена за акцию в рублях. График построен в среде MATLAB.

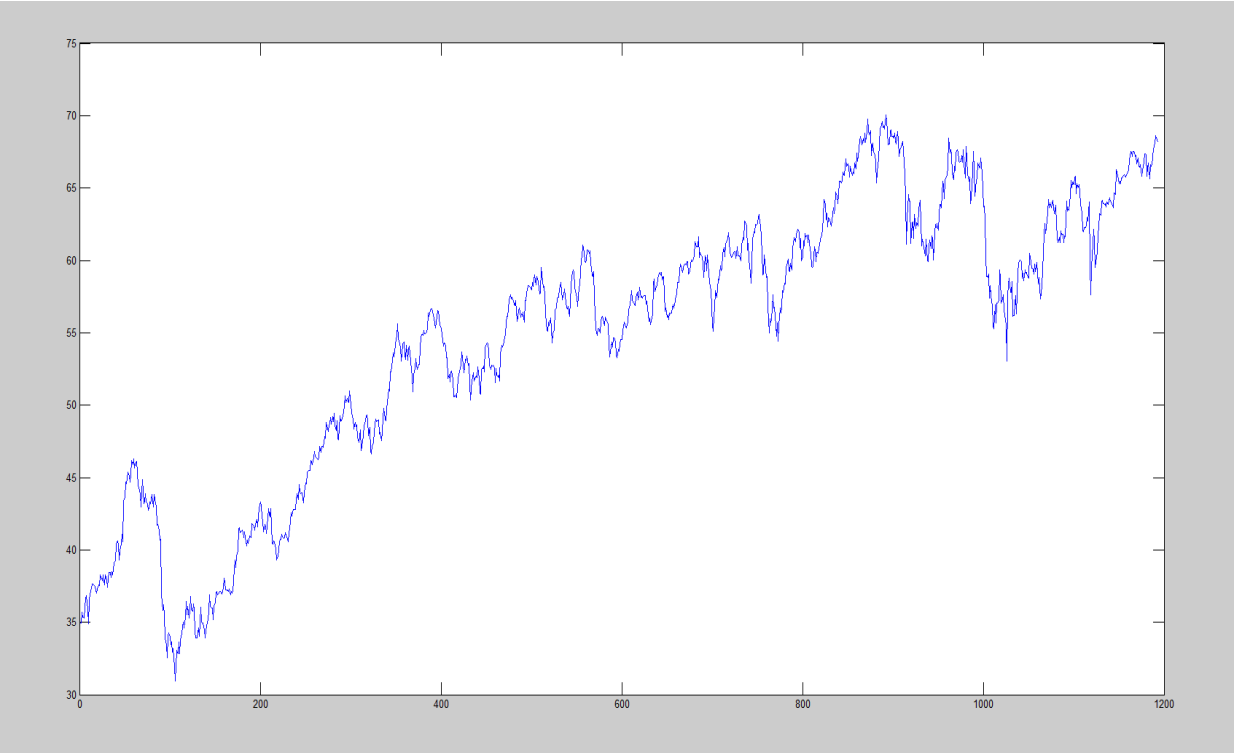


Рисунок 8. Котировки компании JPMorgan Chase с частотой дискретизации в 1 день. На графике по оси x – дневные отсчеты, по оси y – цена закрытия за акцию в долларах США. График построен в среде MATLAB.

Предобработка данных заключалась в применении EMD-разложения исходных рядов и вычитание из них шумовых компонент. Метод был реализован в программе MATLAB, исходный код взят с официального сайта разработчика [26], и представлен в приложении 3 настоящей дипломной работы. Для проведения декомпозиции в MATLAB вызывалась функция:

```
imf = (emd(data,'stop',[0.001,0.005,0.0005],'maxiterations',100))' ;
```

Здесь data - исходный временной ряд.

Поскольку количество мод не определяется методом заранее, было установлено, что при декомпозиции ряда на 10 частотных функций и меньше, для целей фильтрации шума в данных из него будут вычитаться первые 2 моды. В случае, если в результате применения метода EMD образуется больше 10 мод, то для той же цели вычитались первые 3 моды.

Для наглядности на рисунке 9 представлено полное разложение на эмпирические моды временного ряда котировок компании «Аэрофлот».

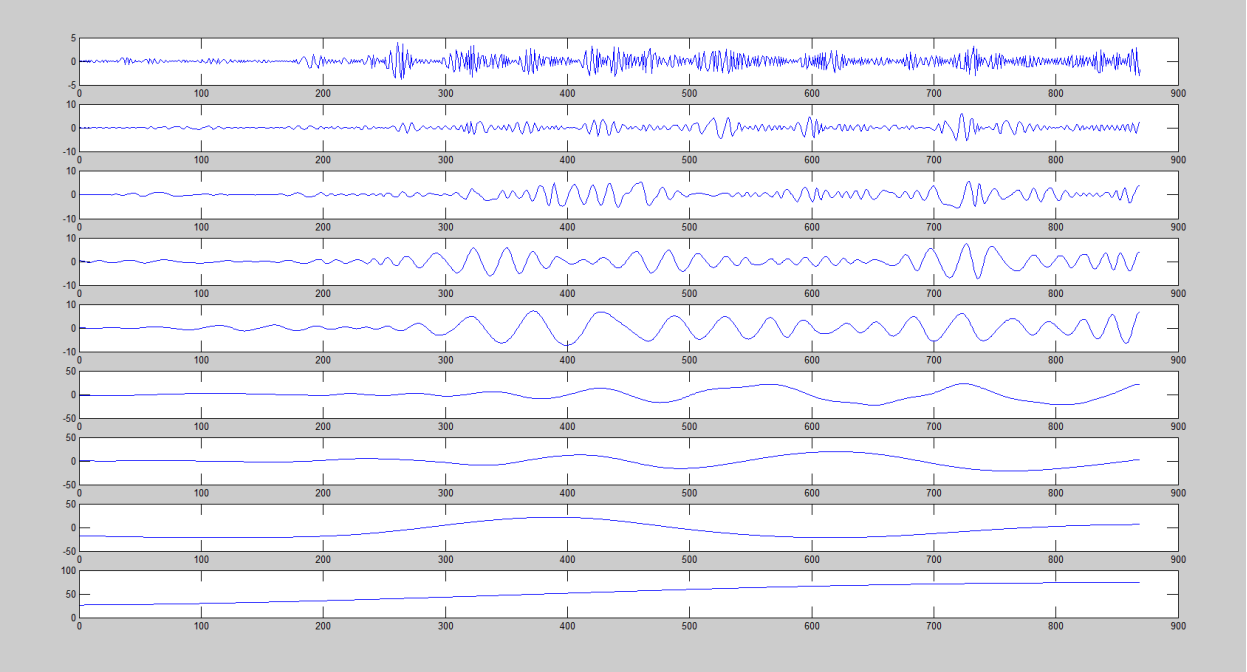


Рисунок 9. Результат EMD-разложения для временного ряда компании «Аэрофлот». На графике представлены 9 мод, для каждой по оси x отложены – дневные отсчеты, по оси y – диапазон колебаний. График построен в среде MATLAB.

Временные ряды, из которых вычтены шумовые компоненты, в дальнейшем изложении практической части будут определяться, как фильтрованные. Все последующие действия производятся с фильтрованными данными.

Второй этап. Формирование обучающей выборки из обработанных данных

Как было изложено в главе 2.5 теоретической части «Формирование входных признаков для обучения нейронной сети», временной ряд задается формулой

$$X = \{x(i) : 0 \le i \le n\} \quad (1)$$

Тогда каждый обучающий пример состоит из входа:

$$X_i = [x(i), x(i-\tau), x(i-2\tau), ..., x(i-m+1)\tau] \quad (2)$$

и соответствующего ему выхода нейросети:

$$Y_i = x(i+1) \quad (3)$$

Для формирования пространства лаговых векторов необходимо вычислить параметры реконструкции m и τ. Временная задержка была вычислена с помощью метода взаимной информации, а размерность пространства вложения – с помощью метода ближайших ложных соседей. Алгоритмы вычисления были реализованы в программе MATLAB, исходные коды были взяты с официального сайта разработчика[27] и представлены в приложениях 1 и 2 настоящей дипломной работы.

Результаты вычислений m и τ для временных рядов котировок акций исследуемых компаний представлены в таблице 1

Финансовый инструмент	Временная задержка, τ	Размерность пространства вложения, m
ПАО «Газпром»	3	7
ПАО «Аэрофлот»	3	6
ПАО «Яндекс»	4	6
ПАО «Сбербанк»	3	7
ПАО «Мегафон»	5	9

«Apple»	4	8
«McDonald’s»	3	8
«JPMorgan&Chase»	4	7

Таблица 1. Результаты вычисления параметров реконструкции для исследуемых финансовых инструментов.

С точки зрения визуального восприятия, механизм погружения – преобразование ряда в матрицу. Практичней, удобней и быстрее сформировать обучающее множество было в программе Excel (рис. 9). Так, каждый элемент множества был наглядно представлен, а работать с отдельными фрагментами общей структуры было значительно удобней, чем если бы это происходило в программе MATLAB.

1																	
2	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695	1696
3																	
4	106,772	108,531	110,418	112,267	113,91	115,23	116,305	117,251	118,135	119,032	119,959	120,697	120,966	120,577	119,711	118,693	118,059
5																	
6	108,531	110,418	112,267	113,91	115,23	116,305	117,251	118,135	119,032	119,959	120,697	120,966	120,577	119,711	118,693	118,059	118,235
7																	
8	106,772	108,531	110,418	112,267	113,91	115,23	116,305	117,251	118,135	119,032	119,959	120,697	120,966	120,577	119,711	118,693	118,059
9	103,892	104,303	105,307	106,772	108,531	110,418	112,267	113,91	115,23	116,305	117,251	118,135	119,032	119,959	120,697	120,966	120,577
10	105,441	104,711	104,064	103,892	104,303	105,307	106,772	108,531	110,418	112,267	113,91	115,23	116,305	117,251	118,135	119,032	119,959
11	104,979	105,628	105,785	105,441	104,711	104,064	103,892	104,303	105,307	106,772	108,531	110,418	112,267	113,91	115,23	116,305	117,251
12	101,125	102,676	103,955	104,979	105,628	105,785	105,441	104,711	104,064	103,892	104,303	105,307	106,772	108,531	110,418	112,267	113,91
13	94,9343	97,2344	99,3073	101,125	102,676	103,955	104,979	105,628	105,785	105,441	104,711	104,064	103,892	104,303	105,307	106,772	108,531
14																	

Рисунок 9. Screenshot рабочего пространства Excel для последних отсчетов временного ряда компании «Аэрофлот». Здесь значения строки 2 соответствуют параметру i из формул (1,2,3), значения строки 4 соответствуют $x(i)$ из формулы (1), значения строки 6 - Y_i из формулы (3), а вектору в строках 8-13 - X_i из формулы (2). Серым цветом выделена тестовое множество, на котором проводилось тестирование уже обученной нейросети.

Третий этап. Обучение нейросети и получение прогнозных данных для каждого финансового инструмента

В тулбоксе NeuralNetwork программы MATLAB существует инструмент nntool, который позволяет создавать, обучать и использовать искусственные нейронные сети. Для открытия пользовательского интерфейса вызывается функция «nntool», открывающая интерактивное окно (рис 10). Ниже подробно описано, как происходило получение прогнозных данных на примере компании «Сбербанк».

Сначала , в рабочее пространство MATLAB загружались данные обучающего множества. В этом примере данные входов нейросети содержатся в объекте «In_sber», выходы - в объекте «Tg_sber», а в объекте «Est_sber» - входные данные для получения первого прогноза.

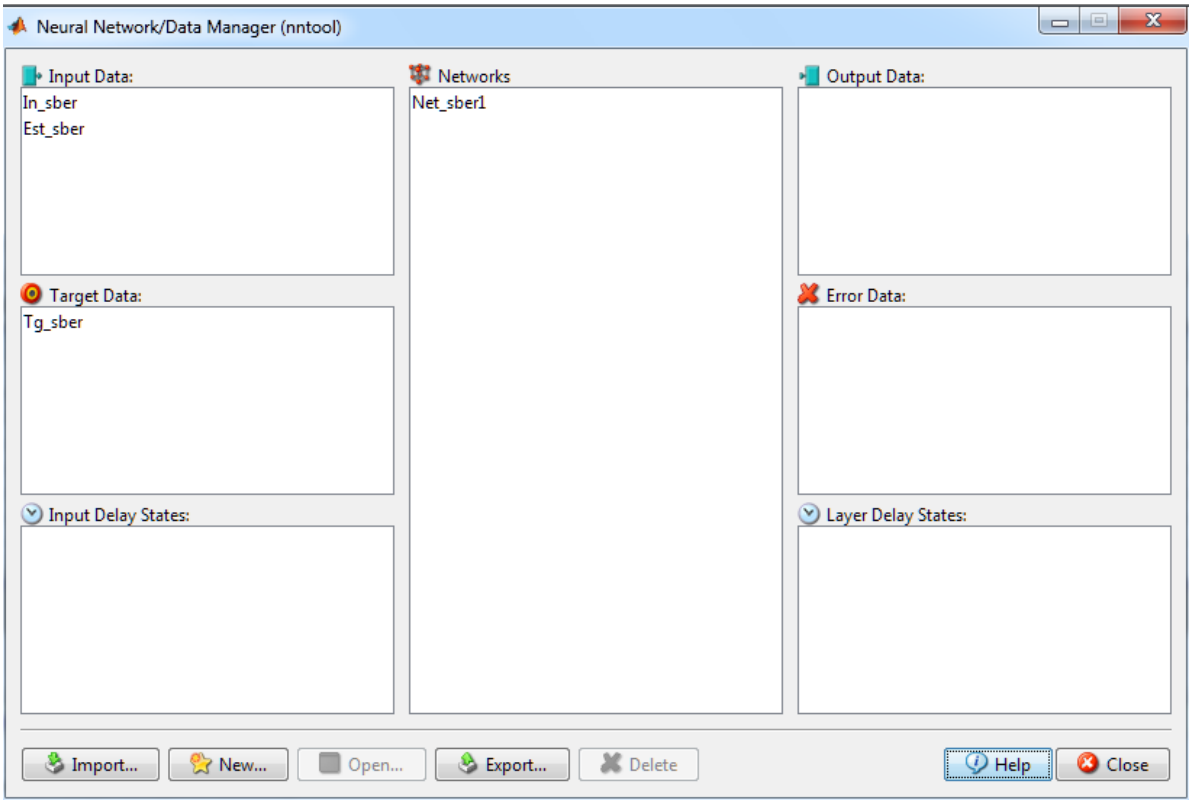


Рисунок 10. Screenshot пользовательского интерфейса инструмента nntool. С помощью клавиш Input/Export можно загружать данные из рабочего пространства Matlab и наоборот. В окне Input Data отображаются элементы обучающего множества, соответствующего входам нейронной сети, Target

Data – выходам. В окне Output Data сохраняется результат, выданный ИНС, а в окне Networks – созданные сети.

Для создания отдельной сети необходимо нажать на клавишу «New», после чего откроется окно, в котором предлагается выбрать параметры новой нейронной сети. Была создана сеть «Net_sber1» со следующими параметрами: архитектура – персептрон с тремя скрытыми слоями по 50 нейронов в каждом, обучение методом Левенберга-Марквардта, оценка функционирования сети – среднеквадратичная ошибка, функция активации – гиперболический тангенс. После выбора параметров нейронная сеть обучается на выбранных данных.

Выбор архитектуры сети с 3 скрытыми слоями по 50 нейронов в каждом обусловлен следующим численным экспериментом: создавалось 5 сетей, отличающихся лишь количеством скрытых слоев и количеством нейронов в них. После обучения всех сетей на одинаковых данных был получен прогноз каждой на один день. Результаты эксперимента представлены в таблице 2.

Архитектура сети	Модуль ошибки	Значение прогноза	Реальное значение
1 скрытый слой, 30 нейронов	3,41	40,61	44,02
2 скрытых слоя, 30 нейронов	0,37	44,39	
2 скрытых слоя, 50 нейронов	0,30	44,33	
3 скрытых слоя, 30 нейронов	0,01	44,03	
3 скрытых слоя, 50 нейронов	0,00	44,02	

Таблица 2. Результаты численного эксперимента для определения архитектуры сети. Обучение производилось на фильтрованных котировках компании «Аэрофлот». Значение прогноза и реальное значение – это цены акций в рублях.

По завершению процесса обучения необходимо было экспортировать сеть в рабочее пространство MATLAB, после чего в командной строке вызывается функция моделирования «sim(net,P)», где net – название ИНС, которую нужно использовать, P – ее входы. Таким образом, получается прогноз на один день.

В данной дипломной работе глубина прогноза для отдельных сетей и комитета была определена в 7 дней. Для того, чтобы получить предсказание на этот период необходимо спрогнозировать на 1 день, после чего полученное значение включается в обучающую выборку, и формируется новый вектор входа для прогноза значения 2-го дня. Процедура повторяется, пока глубина прогноза не достигнет требуемого значения. Визуализация исходных и спрогнозированных данных за период в 1 неделю для компании «Сбербанк» представлен на рисунке 11.

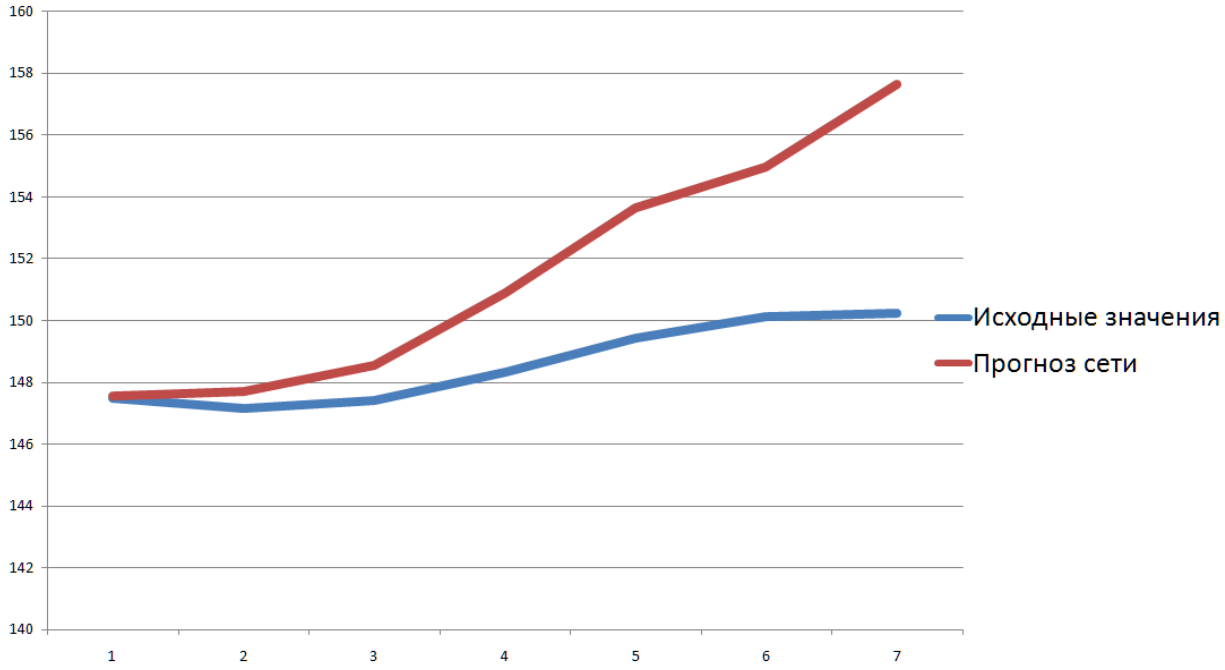


Рисунок 11. Прогноз одной нейросети котировок акций компании «Сбербанк» на 7 дней. На графике по оси x – дневные отсчеты, по оси y – цена за акцию в рублях. График построен в программе Excel. Анализируя график, можно заметить, что сеть обучена не очень хорошо. Прогноз на один

день практически повторяет исходное значение, но сеть не уловила общую динамику, а значит, в длительной перспективе ошибка будет неограниченно возрастать.

Как и для компании «Сбербанк», на этом этапе были получены прогнозные данные для всех финансовых инструментов. Количественная оценка прогноза – показатель среднеквадратичного отклонения (MSE). Вычисляется по формуле:

$$MSE = \frac{1}{N}\sum_{t=1}^{N}(Z(t) - \hat{Z}(t))^2$$

Здесь: $Z(t)$ - фактическое значение временного ряда, $\hat{Z}(t)$ - прогнозное.

Визуализации прогнозов отдельных сетей представлены на рисунках 12-19 приведенных ниже. Показатель MSE для предсказанных значений каждого финансового инструмента представлен в таблице 3., приведенной ниже.

Четвертый этап. Формирование комитета искусственных нейронных сетей и получение прогноза

На заключительном этапе вычислений требовалось создать комитет ИНС, с помощью которого можно было эффективно прогнозировать исследуемые финансовые инструменты. Таким образом, в постановке задачи делался акцент на многообразии паттернов, которые должен идентифицировать комитет. Было создано 50 нейронных сетей с различной архитектурой, функциями активации, в большей степени различавшихся обучающими выборками. По завершению процесса обучения формировался комитет ИНС. Принцип отбора сетей в него был следующий: строился прогноз на один день, после чего для каждой сети считался показатель MSE. По результатам вычисления из общего списка выбрасывалось 5 сетей с худшими показателями. Потом строился прогноз на следующий день, и по тому же принципу отсеивались еще 5 сетей. Таким образом, после получения

недельного прогноза в комитет вошли 15 сетей с наилучшими показателями эффективности.

Такой принцип отбора был обусловлен тем, что в большинстве предшествующих прогнозов на срок больше 1 дня выделялась следующая очевидная закономерность: с каждым последующим отсчетом возрастала ошибка сети. Остальные случаи, когда ошибка сначала возрастала, затем убывала, являлись следствиями заучивания нейронной сетью случайных или неверных закономерностей, поскольку в длительной перспективе после смены направления тренда ошибка начинала сильно возрастать. Применяя такой алгоритм отбора, сразу отсеивались наименее эффективные сети, а также сокращалось общее время формирования комитета.

Решения по комитету принималось в соответствии с пятым способом, описанным в главе 2.7 теоретической части настоящей дипломной работы. В качестве ответа по комитету принималось значение прогноза той сети, которая на момент времени t имела наименьшее значение по критерию суммы квадратов ошибки прогнозов [25], рассчитывающийся по формуле:

$$K(i) = \sum_{s=1}^{k} ((y_i)(t-s) - y_{\hat{i}}(t-s))^2$$

Здесь t - текущий временной отсчет, k - количество спрогнозированных отсчетов до момента времени t, $y_{\hat{i}}(t-s)$ - фактическое значение ряда в момент времени $(t-s)$, $y_i(t-s)$ - прогнозное значение ряда, а i - номер сети в комитете.

Рассмотрим более подробно результаты прогноза каждого финансового инструмента.

1. Прогноз котировок компании «Аэрофлот» (Рис 12.).

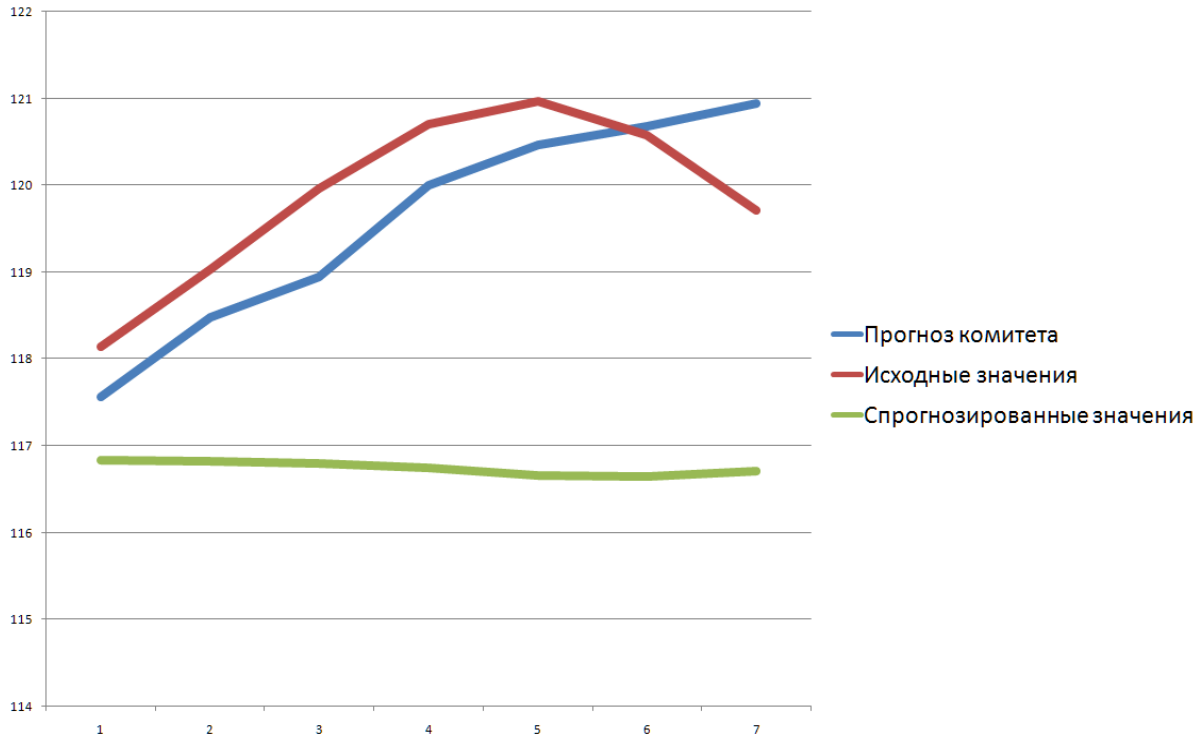


Рисунок 12. Прогноз котировок компании «Аэрофлот». На графике по оси x – дневные отсчеты, по оси y – цена за акцию в рублях. Здесь красным цветом – исходное значение, зеленым – прогноз сети, синим – прогноз комитета. На графике видно, что на первых шести отчетах направление тренда комитета совпадает с исходным трендом, после чего ошибка начинает возрастать.

2. Прогноз котировок компании «Газпром» (рис.13).

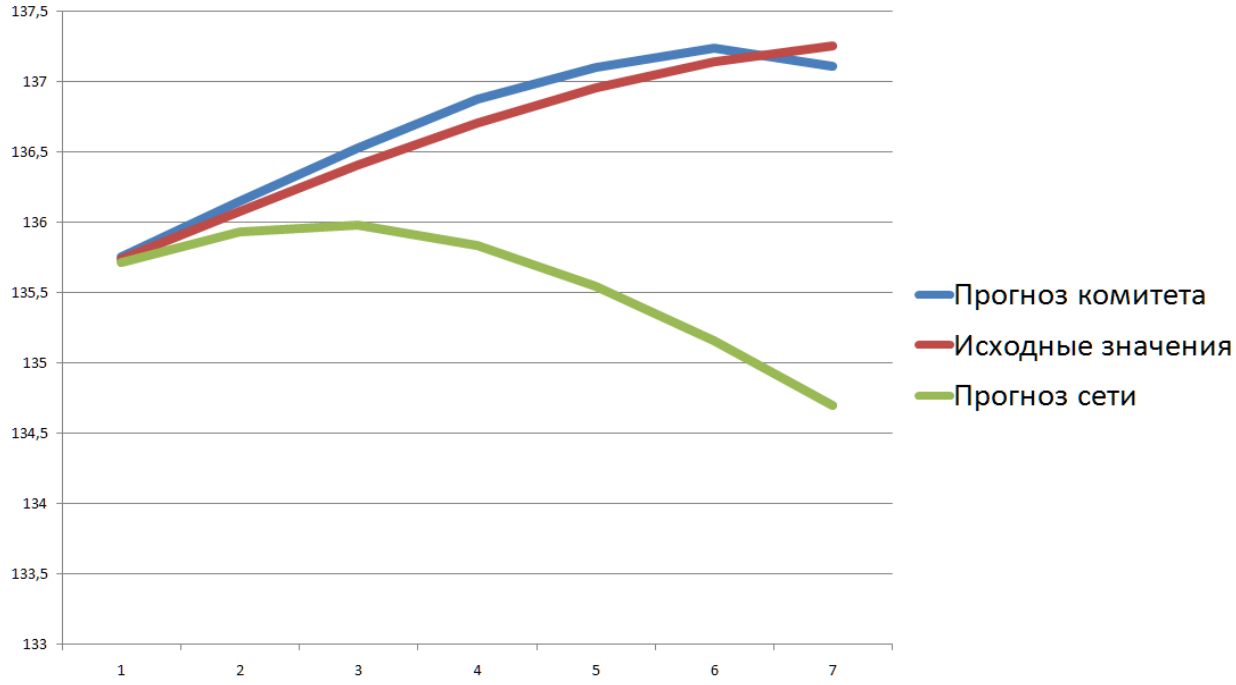


Рисунок 13. Прогноз котировок компании «Газпром». На графике по оси x – дневные отсчеты, по оси y – цена за акцию в рублях. Здесь красным цветом – исходное значение, зеленым – прогноз сети, синим – прогноз комитета. На графике видно, на первых 7-ти отсчетах прогноз комитета практически повторяет динамику цен.

3. Прогноз котировок компании «Мегафон» (рис. 14)

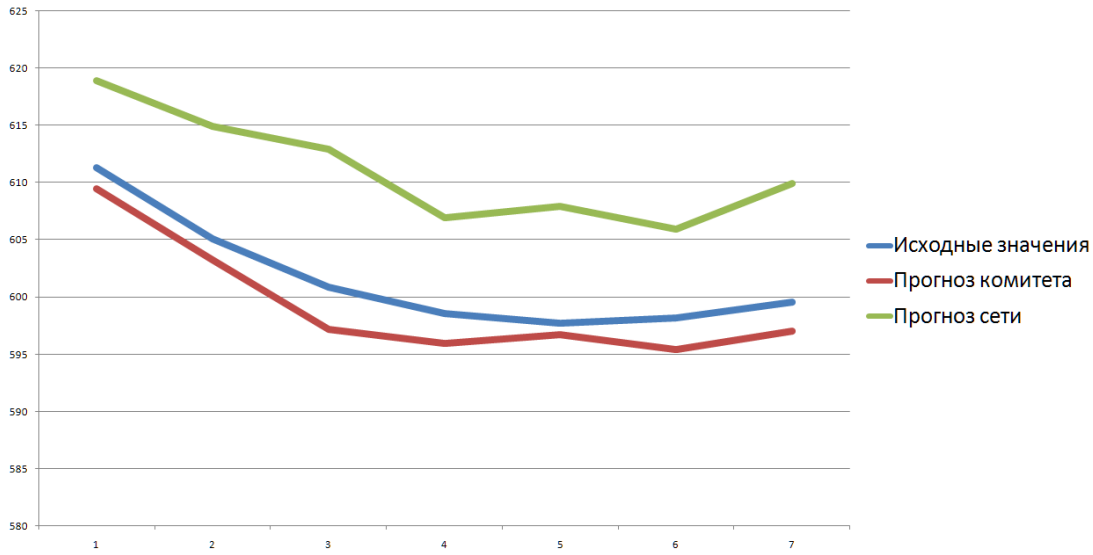


Рисунок 14. Прогноз котировок компании «Мегафон». На графике по оси x – дневные отсчеты, по оси y – цена за акцию в рублях. Здесь синим цветом – исходное значение, зеленым – прогноз сети, красным – прогноз комитета. На графике видно, что на всех 7 отсчетах прогноз комитета повторяет динамику цен. В то же время, прогноз сети хоть и с большей погрешностью, но также улавливает направление тренда.

4. Прогноз котировок компании «Сбербанк» (рис. 15).

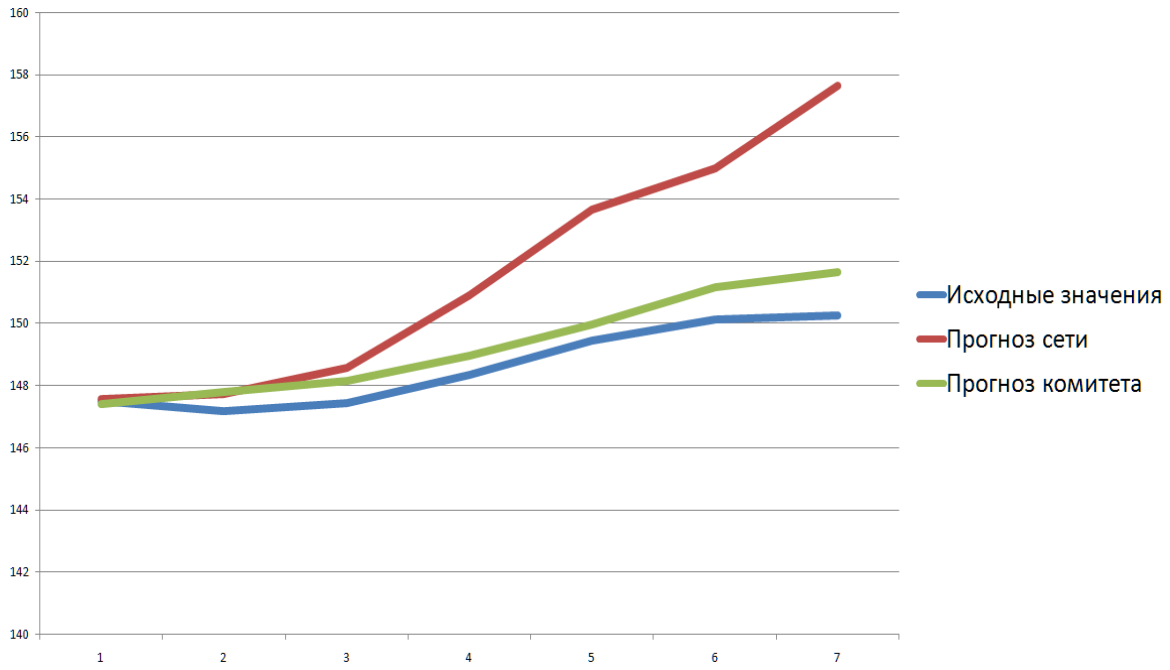


Рисунок 15. Прогноз котировок компании «Сбербанк». На графике по оси x – дневные отсчеты, по оси y – цена за акцию в рублях. Здесь синим цветом – исходное значение, зеленым – прогноз сети, красным – прогноз комитета. На графике видно, что на 7 отсчетах прогноз комитета повторяет динамику цен, в то время как ошибка сети начинает резко возрастать с 3-го отсчета.

5. Прогноз котировок компании «Яндекс» (рис.16).

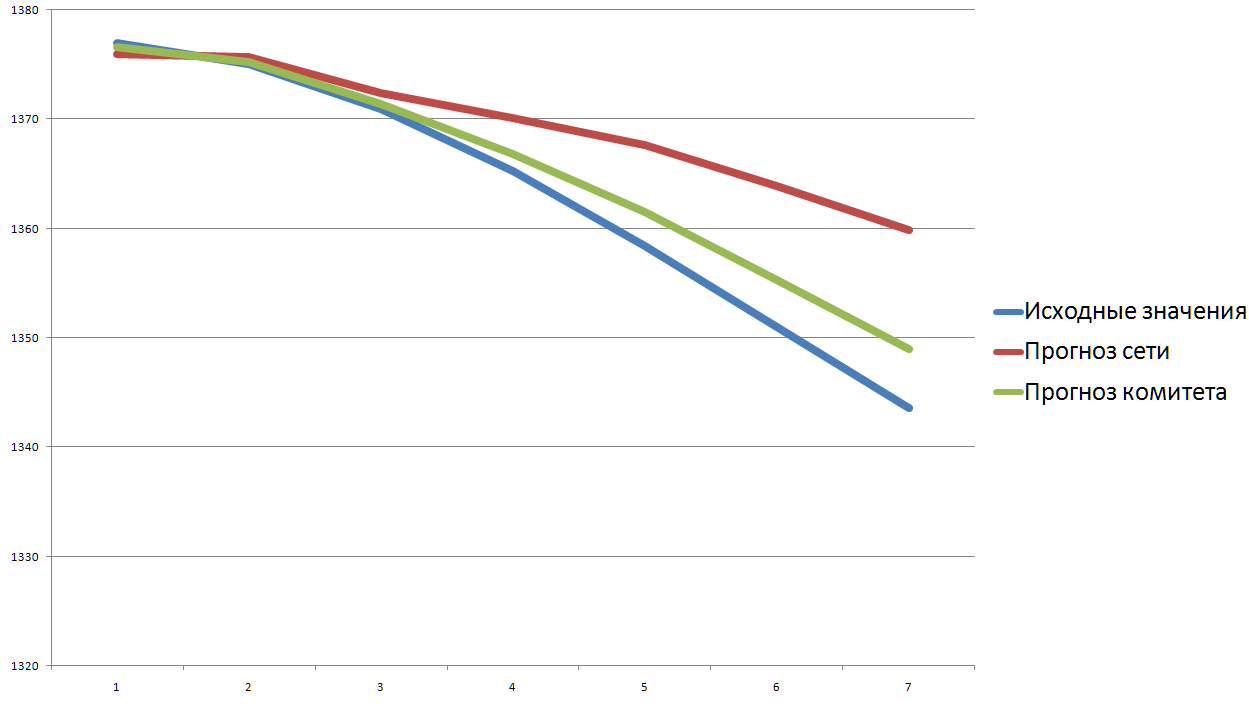


Рисунок 16. Прогноз котировок компании «Яндекс». На графике по оси x – дневные отсчеты, по оси y – цена за акцию в рублях. Здесь синим цветом – исходное значение, красным – прогноз сети, зеленым – прогноз комитета. На графике видно, что на протяжении 7 отсчетов прогноз комитета и прогноз сети повторяют направление тренда исходных значений, но ошибка возрастает с каждым последующим отсчетом.

6. Прогноз котировок компании «JPMorgan&Chase» (рис. 17)

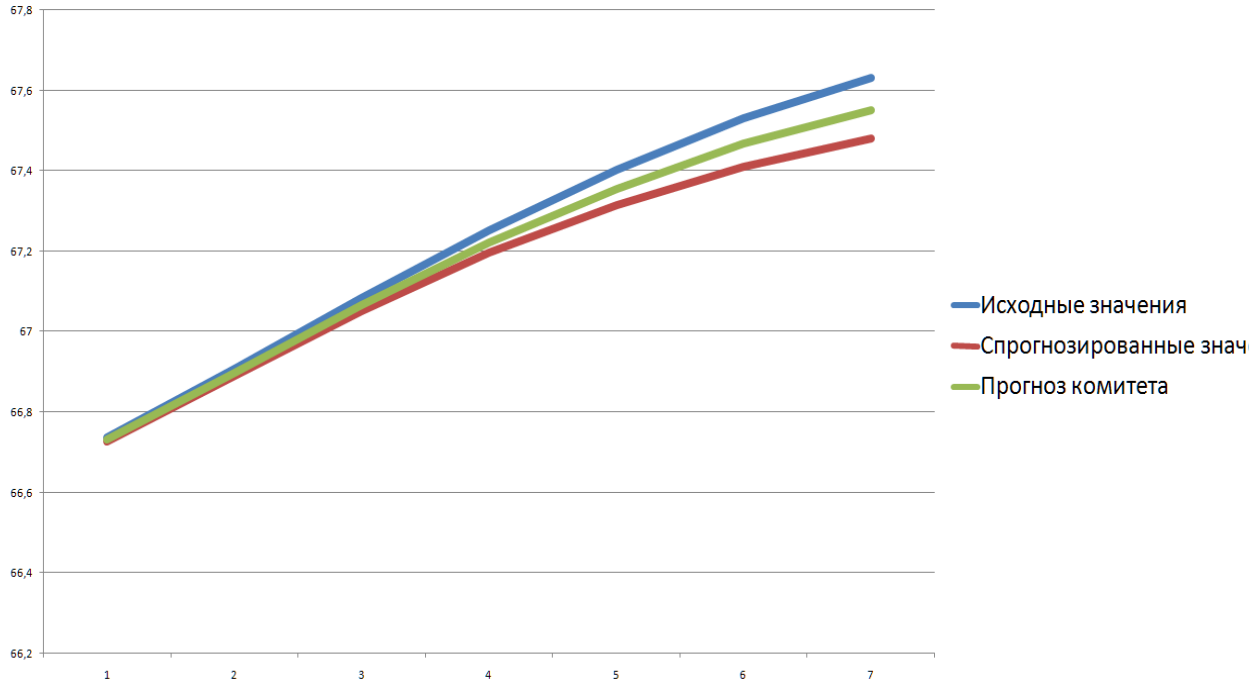


Рисунок 17. Прогноз котировок компании «JPMorgan&Chase». На графике по оси x – дневные отсчеты, по оси y – цена за акцию в долларах США. Здесь синим цветом – исходное значение, красным – прогноз сети, зеленым – прогноз комитета. На графике видно, что на протяжении всех 7-ти отсчетов прогноз комитета и отдельной сети повторяет динамику цен, но при этом у комитета ошибка меньше.

7. Прогноз котировок компании «Apple» (рис.18)

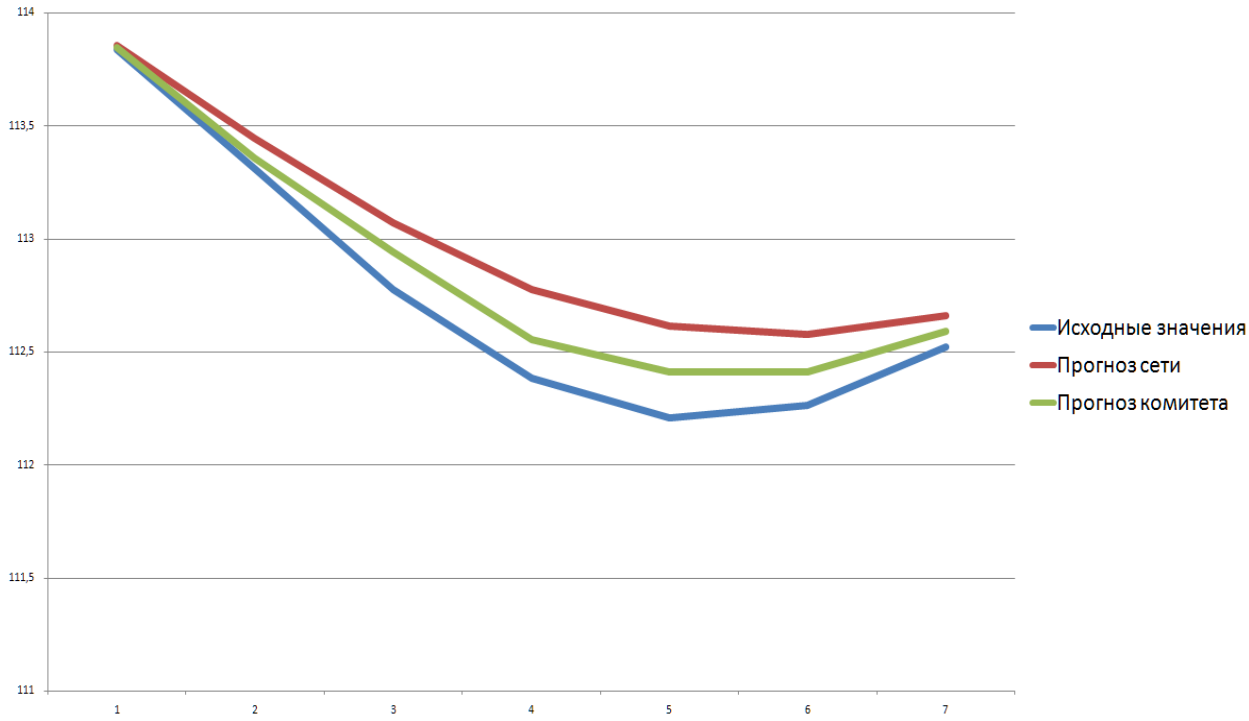


Рисунок 18. Прогноз котировок компании «JPMorgan&Chase». На графике по оси x – дневные отсчеты, по оси y – цена за акцию в долларах США. Здесь синим цветом – исходное значение, красным – прогноз сети, зеленым – прогноз комитета. Можно сказать что, и сеть и комитет уловили основную динамику цен.

8. Прогноз котировок компании McDonald’s (рис.19).

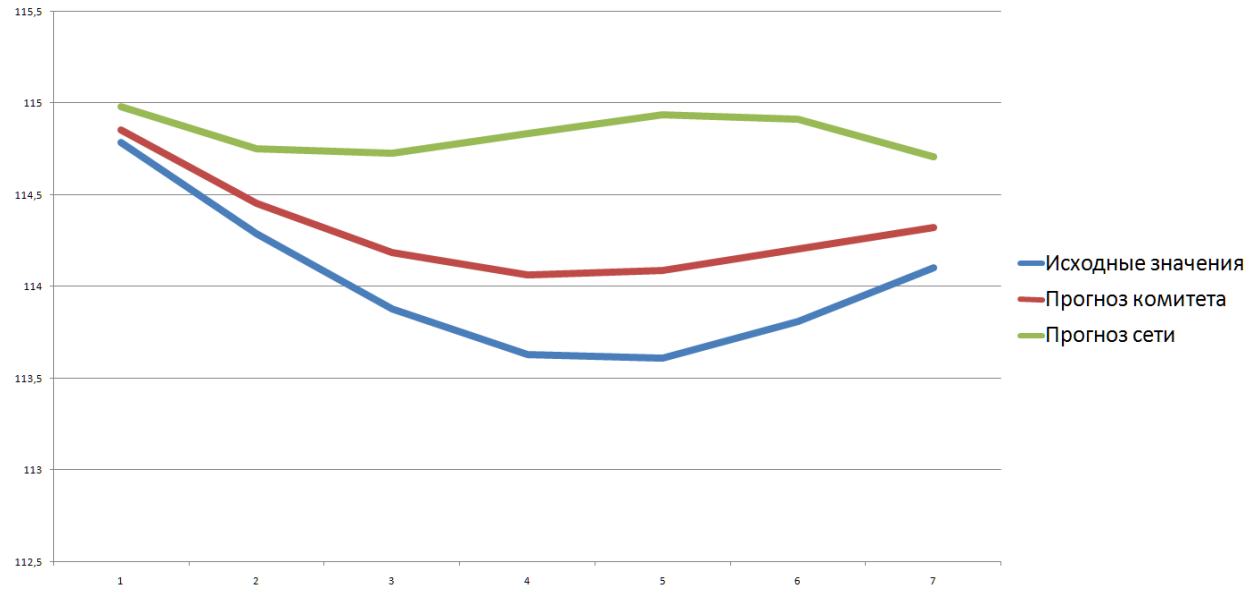


Рисунок 19.Прогноз котировок компании «JPMorgan&Chase». На графике по оси x – дневные отсчеты, по оси y – цена за акцию в долларах США. Здесь синим цветом – исходное значение, красным – прогноз комитета, зеленым – прогноз отдельной сети.

Компания	MSE отдельной сети	MSE комитета
ПАО «Газпром»	1,91	0,02
ПАО «Аэрофлот»	10,71	0,56
ПАО «Яндекс»	78,12	8,68
ПАО «Сбербанк»	14,93	2,1
ПАО «Мегафон»	91,72	5,87
«Apple»	1,17	0,48
«McDonald’s»	0,82	0,11
«JPMorgan&Chase»	0,01	0,00

Таблица 3. Сопоставление коэффициента MSEотдельной сети и комитета для всех финансовых инструментов.

Следует отметить, что показатель MSEможно применить только для оценки качества разных прогнозов одного инструмента. Он дает информацию о величине ошибки, но не о том, насколько велика погрешность относительно реального значения котировки. Коэффициент средней абсолютной ошибки в процентах позволит оценить эффективность прогнозов комитета и отдельных сетей, но и сравнить их между собой. Результаты приведены в таблице 4. Формула расчета коэффициента:

$$MAPE = \frac{1}{N}\sum_{t=1}^{N}\frac{|Z(t)-\hat{Z}(t)|}{Z(t)} * 100\%$$

Компания	MAPE отдельной сети	MAPEкомитета
ПАО «Газпром»	0,77%	0,08%
ПАО «Аэрофлот»	2,59%	0,55%

ПАО «Яндекс»	0,49%	0,16%
ПАО «Сбербанк»	5,57%	1,00%
ПАО «Мегафон»	1,57%	0,38%
«Apple»	0,21%	0,10%
«McDonald’s»	0,72%	0,25%
«JPMorgan&Chase»	0,10%	0,05%

Таблица 4. Сопоставление коэффициента MAPEотдельной сети и комитета для всех финансовых инструментов.

Из таблицы видим, что MAPE комитета по всем финансовым инструментам не превышает 1%. Это еще раз подтверждает, что для получения высококачественных прогнозов использование комитетов ИНС неизбежно.

Пятый этап. Анализ полученных результатов.

Получив результаты нейропрогнозирования, представив их визуально и оценив количественно, можно сделать 2 основных вывода:

1. При решении задачи прогнозирования финансовых временных рядов эффективней использовать комитеты, нежели отдельные сети. Применение комитетов улучшило качество прогнозов в среднем на 37%

2. Погрешность прогноза акций американских компаний оказалась в среднем на 30% меньше, чем отечественных. Такое различие могло явиться следствием того, что американский фондовый рынок значительно старше и стабильней отечественного. Таким образом, механизмы внутреннего функционирования являются более сформировавшимися и более трендоустойчивыми. Таким образом, нейросеть лучше выучивает скрытые закономерности. Из этого можно сделать вывод о том, что,

основываясь на данной методологии, строить торговую стратегию более эффективно на американском фондовом рынке, чем на отечественном.

Заключение

В заключение следует сказать, что применение нейронных сетей в области прогнозирования финансовых временных рядов становится все более широким, а методология – более обширной. Это подтверждается ежегодно растущим количеством исследований и статей, посвященных различным способам нейропрогнозирования.

Результаты применения ИНС для предсказания курса акций в данной дипломной работе показали, что хорошо обученная сеть может проводить анализ временного ряда котировок и осуществлять предсказания с различным уровнем точности. Основываясь на численных данных, полученных в результате вычислений в данной дипломной работе можно сказать, что описанная модель пригодна для прибыльной торговли на рынке, если добиться фиксированной и приемлемой для инвестора точности прогнозирования, которая, как известно не должна превышать в абсолютных величинах 0.001 от единицы измерения цен финансовых инструментов. В свою очередь, результаты экспериментов показали, что точность прогнозирования зависит от многих факторов, например, предобработки данных и выбора архитектуры нейросети, а также может быть улучшена с помощью применения комитета ИНС. Из этого можно заключить, что работа по созданию собственной торговой стратегии имеет перспективы и может быть продолжена.

Несмотря на широкое применение нейросетевых технологий в анализе финансовых данных стоит отметить, что алгоритмы предсказания носят, в большей степени, экспериментальный характер. Комитеты ИНС способны

улавливать различные взаимодействия и нелинейные корреляции, но перед тем, как применять данные прогнозов для совершения сделок на бирже, необходимо провести дополнительные исследования с целью строгой формализации факторов, определяющих точность прогноза.

Библиография

1. Едронова В.Н., Мизиковский Е.А. Учет, оценка доходности и анализ финансовых вложений: учебное пособие. 2-е изд, доп. Москва: Магистр, 2011. 364 с.
2. Стародубцева Е.В. Рынок ценных бумаг: Учебник. Москва: ИД «ФОРУМ»: ИНФРА-М , 2006. —176 с.
3. Индекс голубых фишек RTSSTD [Электронный ресурс] // Сайт московской биржи. URL: http://moex.com/ru/index/RTSSTD (Дата обращения 16.05.2017)
4. Дегтярева О.И. Биржевое дело: Учебник. Москва: Магистр, 2007.624 с.
5. Котировки акций РФ и США [Электронный ресурс] // Финам – инвестиционная компания. URL: https://www.finam.ru/analysis/quotes/ (Дата обращения 16.05.2017)
6. Huang, et al, The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis.// Proc. R. Soc. Lond. A. // 1998.vol. 454, pp. 903—995.
7. Кратович П. В. Предпрогнозный анализ временных рядов финансовых данных на основе методов фрактального анализа // Молодой ученый. 2010. №1-2. Т. 1. С. 11-18.
8. Макаренко Н.Г. Эмбедология и нейропрогноз // Научная сессия МИФИ-2003. V Всероссийская научно-техническая конференция «Нейроинформатика–2003»: Лекции по нейроинформатике. Часть 1. – Москва: МИФИ, 2003. С. 86-141.

9. Грицай А.А. Определение эффективной структуры входных данных для обучения нейросети решению задачи прогнозирования спроса. // Вестник Тверского Государственного Университета. Серия: Прикладная математика. 2014 №2. С. 95-106.

10. Головко В.А. Нейросетевые методы обработки хаотических процессов // Научная сессия МИФИ-2005. VII Всероссийская научно-техническая конференция «Нейроинформатика–2005»: Лекции по нейроинформатике. – Москва: МИФИ, 2005. С. 43-91.

11. Рюэль Д. Случайность и хаос. – М.: Издательство «Регулярная и хаотическая динамика», 2001. – 192 с.

12. Рюэль Д., Такенс Ф.О. природе турбулентности. Странные аттракторы. Москва: Мир, 1981.– 117 – 151 с.

13. TakensF. Detecting Strange Attractors in Turbulence. // Lecture Notes in Mathamatics. 1981. Vol. 898, Springer-Verlag, pp. 366-381.

14. Lorenz E.N. Deterministic nonperiodic flow. // J. Atmos. Sci. 1963. Vol.20, pp. 130-141.

15. Малинецкий Г.Г., Потапов А.Б., Подлазов А.В. Нелинейная динамика: Подходы, результаты, надежды. Изд. 3-е. – Книжный дом «ЛИБРОКОМ», 2011. – 280с.

16. Лоскутов А. Ю., «Очарование хаоса», Успехи Физических Наук, 2010. Том 180. № 12. с.1305-1329

17. Мак-Каллок У.С., Питтс В., Логическое исчисление идей, относящихся к нервной активности // В сб.: «Автоматы» под ред. К.Э. Шеннона и Дж. Маккарти. — Москва: Изд-во иностр. лит., 1956. — с.363-384. (Перевод английской статьи 1943г.)

18. Галушкин А. Нейронные сети. История развития теории. Учебное пособие. Москва: Альянс, 2016.

19. Горбань А. Н., Нейроинформатика: кто мы, куда мы идем, как путь наш измерить // Вычислительные технологии. — Москва: Машиностроение. 2000. №4. — С. 10-14.

20. Хайкин С. Нейронные сети: полный курс, 2-е издание. Пер. с англ. – Москва: Вильямс, 2006. – 1104 с.

21. Ежов А.А., Шумский С.А. Нейрокомпьютинг и его применения в экономике и бизнесе (серия "Учебники экономико-аналитического института МИФИ" под ред. проф. В.В. Харитонова). Москва: МИФИ, 1998. - 224 с.

22. Терехов С.А., Научная сессия МИФИ–2007. IX всероссийская научно-техническая конференция «Нейроинформатика–2007»: Лекции по нейроинформатике. Часть 2. – Москва: МИФИ, 2007. – 148 с.

23. Хайкин С. Преимущества и ограничения обучения методом обратного распространения // Нейронные сети. – Москва: Вильямс, 2006. – с. 304–314.

24. Пархоменко С. С., Леденёва Т. М. Обучение нейронных сетей методом Левенберга-Марквардта в условиях большого количества данных. //Вестник ВГУ, Серия: Системный анализ и информационные технологии, 2014, № 2. – с. 98–106.

25. Каширина И.Л. О методах формирования нейросетевых ансамблей в задачах прогнозирования финансовых временных рядов. // Вестник ВГУ, Серия: Системный анализ и информационные технологии, 2009, № 2.— с. 114-117

26. Fileexchange [Электронный ресурс]// Сайт компании Mathworks.URL: https://www.mathworks.com/matlabcentral/fileexchange/?s_tid=gn_mlc_fx (Дата обращения 16.05.17).

Приложение 1. Программная реализация метода взаимной информации.

```
% Estimation of optimal lag for arbitrary time series by means of the
```

```
% mutual information function
function [lag,S,elapsedTime]=mutualOptLag(x,bins,maxLag)
% Input arguments:
% - x - time series vector
% - bins - number of boxes
% - maxLag - maximum time delay(lag)
% Output arguments^
% - lag - optimal lag, which is defined as a first minimum of mutual
% information function
% - S - vector contains values of mutual information function at integer
% points
% elapsedTime - time of estimation in seconds
startTime=cputime;


% Length of x
lx=length(x);


% Transposition of x, if it is column-vector
ss=size(x);
if ss(1)>1,
    x=x';
end


% bins and maxLag are optional arguments. If they are not declared by a
% user, they are defined with the default values
if nargin<3,
    maxLag=100;
end
if nargin<2,
    maxmin_range = max(x)-min(x);
    bins = ceil(maxmin_range/(2.0*iqr(x)*lx^(-1/3)));


end


% Estimation of mutual information function with different lags
S=zeros(1,maxLag);
for tau=1:maxLag,
    S(tau)=mutual(x,bins,tau);
end


% Optimal lag is the first minimum of mutual information function
lag=minimum(S);


% Time of estimation
elapsedTime=cputime-startTime;


%--------------------------------------------------------------------------
% Subfunctions
%--------------------------------------------------------------------------


%----------------------------------------------------------------
% Mutual information function
function S=mutual(x,bins,tau)
S=0;


% "partition" is a function, returning bins of x range partition
part=partition(x,bins);
% "probability" is a vector of probabilities of finding x(t) value in i-interval of
% partition "part"
[prob,n]=probability(x,part);
% "jProb" is a matrix, each element (i,j) is a joint probability of finding x(t)
% value in i-interval and x(t+tau) value in j-interval of partition "part" of time
% series "x"
jProb=jointProb(x,part,tau);


for i=1:bins,
for j=1:bins,
if jProb(i,j)~=0,
        S=S+jProb(i,j).*log(jProb(i,j)./(prob(i).*prob(j)));
end
end
end
%----------------------------------------------------------------


%----------------------------------------------------------------
% Partition function - returns bins of time series x partition
function part=partition(x,bins)
% part is a (bins x 2)-matrix, each row contains left and right bounds of
% bin of x partition


% step (or range) of partition
```

```
mm=minmax(x);
interval=mm(2)-mm(1);
step=interval/bins;


part=zeros(bins,2);
for i=1:bins,
    part(i,1)=mm(1)+ (i-1)*step;
    part(i,2)=mm(1)+ i*step;
end
%---------------------------------------------------------------------


%---------------------------------------------------------------------
% Probability function
function [prob,n]=probability(x,part)
% prob is a (1 x bins)vector, containing probabilities of finding x(t) in i-bin
% n is a (1 x bins)vector, containing number of points in i-bin
ss=size(part);
bins=ss(1);
prob=zeros(1,bins);
n=zeros(1,bins);
lx=length(x);
mm=minmax(x);



for i=1:bins,
% forming n(i) - the number of points x(t) belonging to i-bin
for j=1:lx,
if x(j)>=part(i,1) & x(j)<part(i,2),
           n(i)=n(i)+1;
end
if i==bins & x(j)==mm(2),
           n(i)=n(i)+1;
end
end
    prob(i)=n(i)/lx;
end
%---------------------------------------------------------------------


%---------------------------------------------------------------------
% Joint probability function
function jProb=jointProb(x,part,tau)
% jProb is a (bins x bins)-matrix, each element (i,j) is a joint probability of finding x(t)
% value in i-interval and x(t+tau) value in j-interval of partition "part" of time
% series "x"


ss=size(part);
bins=ss(1);
jProb=zeros(bins,bins);
n=zeros(bins,bins);
lx=length(x);
mm=minmax(x);


for i=1:bins,
for j=1:bins,
for k=1:(lx-tau),
if x(k)>=part(i,1) & x(k)<part(i,2) & x(k+tau)>=part(j,1) & x(k+tau)<part(j,2),
               n(i,j)=n(i,j)+1;
end
% additioanally checking for the boundary of interval
if i==bins & x(k)==mm(2) & x(k+tau)>=part(j,1) & x(k+tau)<part(j,2),
               n(i,j)=n(i,j)+1;
end
if  x(k)>=part(i,1) & x(k)<part(i,2) & j==bins & x(k+tau)==mm(2),
               n(i,j)=n(i,j)+1;
end



if i==bins & x(k)==mm(2) & j==bins & x(k+tau)==mm(2),
               n(i,j)=n(i,j)+1;
end


end
% Remind: joint probability equals p(AB)=n(AB)/n
        jProb(i,j)=n(i,j)/(lx-tau);
end
end
%---------------------------------------------------------------------


%---------------------------------------------------------------------
% Position of First minimum of row-vector x
function m=minimum(x)
% m=0 if x has no minimum
lx=length(x);
```

```
less=zeros(1,lx-1);



for i=1:(lx-1),
if x(i+1)<x(i),
        less(i)=1;
end
end


mm=1;
i=1;
while less(i)==1,
    i=i+1;
    mm=i;
if i==lx-1,
        less(i)=0;
end
end


if mm>=1,
    m=mm;
else
    m=0;
end
%---------------------------------------------------------------------
```

Приложение 2. Программная реализация алгоритма ближайших ложных соседей.

```
% Estimation of embedding dimension for arbitrary time series with known
% lag by means of false nearest neighbors(FNN) method
function [embDim,fnnPer,elapsedTime]=fnnEmbDimImp(x,lag,tresh)
% Input arguments:
% - x - time series vector
% - lag - optimal lag for x
% - maxDim - maximal embedding dimension
% - tresh - treshold R
% Output arguments:
% - embDim - optimal embedding dimension, determined as the dimension,
% provided zero percentage of false nearest neighbors
% - fnPer(dim) - vector of percentage of false nearest neighbors at
% different dim
% elapsedTime - time of estimations


startTime=cputime;


% Transposition of x, if it is column-vector
ss=size(x);
if ss(1)>1,
    x=x';
end



% tresh,maxDim are optional arguments. If they are not declared by a
% user, they are defined with the default values


if nargin<3,
    tresh=10;
end


format long;


% Estimation of percentages of FNN at different dimensions
i=1;
fnnPer(1)=falseNearest(x,lag,1,tresh);
while fnnPer(i)~=0,
    i=i+1;
    fnnPer(i)=falseNearest(x,lag,i,tresh);
end
% embDim is equal the dim wich provides zero value of fnnPer
embDim=length(fnnPer);


% Time of estimation
```

```
elapsedTime=cputime-startTime;

%-------------------------------------------------------------------------
% Subfunctions
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
% falseNearest function
function fnnPer=falseNearest(x,lag,dim,tresh)
% Return the percentage of FNN at given embedding dim

% Building a embedding into dim-dimensional lag-space
lx=length(x);
% Amount of lag vectors
tmp=lx-lag*dim;

% Massive of "extended" lag vectors(each i column is dim-dimensional
% lag-vector X(i) and x(i+dim*lag) component of (dim+1)-dimensional lag-vector X(i))
lagExt_v=zeros(dim+1,tmp);
for i=1:tmp,
    lagExt_v(1:dim,i)=x(i : lag : i + (dim - 1) * lag);
    lagExt_v(dim+1,i)=x(i + dim*lag);
end

% Searching for NN-point (nnExt_v) for every lag-point
% nnExt_v - massive of vectors of X(i)-NN
[nnExt_v,nnNorm]=nearest(lagExt_v);



% Determine, if nn is false
N=0;
R=zeros(1,tmp);
for i=1:tmp,
if nnNorm(i)~=0,
        R(i)=abs(lagExt_v(dim+1,i)-nnExt_v(dim+1,i))/nnNorm(i);
% If R>tresh, then i-point has a FNN
if R(i)>tresh,
           N=N+1;
end
end
end
fnnPer=N/tmp;

%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
% Nearest neighbor indixes for all lag-vectors
function [nnExt_v,nnNorm]=nearest(lagExt_v)
% Matrix d(i,j)=||x(i)-x(j)|| is symmetric
lag_v=lagExt_v(1:(end-1),:);
ss=size(lagExt_v);
lv=ss(2);
dist=zeros(lv,lv);
for i=1:lv,
for j=i+1:lv,
        dist(i,j)=norma(lag_v(:,i)-lag_v(:,j));
end
end
% Transpositioning og matrix dist
distT=dist';
norms=dist+distT;

% sorting rows of symmetric matrix norms in ascending order
[sortNorms,nnInd]=sort(norms,2);
% the first column of matrix sortNorms contains zeros from diagonal of martrix norms
% thats why minimum value in norms row is contained in second column of
% sortNorms

nnExt_v=zeros(ss(1),ss(2));
nnNorm=zeros(1,ss(2));
% nnExt_v - is a massive of an extended nearest vectors to each
% lagExt-vector
for i=1:lv,
    nnExt_v(:,i)=lagExt_v(:,nnInd(i,2));
    nnNorm(1,i)=sortNorms(i,2);
end
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
% Length or norm of vector x in R^n space
function n=norma(x)
% for matrix argument returns row vector of norms of matrix columns
```

```
lx=length(x(1,:));
for k=1:lx,
    n(1,k)=(sum(x(:,k).^2))^0.5;
end
%-------------------------------------------------------------------------
```

Приложение 3. Программная реализация метода EMD.

```
function [imf,ort,nbits] = emd(varargin);


[x,t,sd,sd2,tol,display_sifting,sdt,sd2t,ner,nzr,lx,r,imf,k,nbit,NbIt,MAXITERATIONS,FIXE,FIXE_H,MAXMODES,INTERP
,mask] = init(varargin{:});


if display_sifting
  figure
end


% maximum number of iterations
% MAXITERATIONS=2000;



%main loop : requires at least 3 extrema to proceed
while ~stop_EMD(r) & (k < MAXMODES+1 | MAXMODES == 0) & ~any(mask)


% current mode
    m = r;



% mode at previous iteration
    mp = m;



if FIXE
        [stop_sift,moyenne] = stop_sifting_fixe(t,m,INTERP);
elseif FIXE_H
        stop_count = 0;
        [stop_sift,moyenne,stop_count] = stop_sifting_fixe_h(t,m,INTERP,stop_count,FIXE_H);
        stop_count = 0;
else
        [stop_sift,moyenne] = stop_sifting(m,t,sd,sd2,tol,INTERP);
end


if (max(m) - min(m)) < (1e-10)*(max(x) - min(x))
if ~stop_sift
            warning('forced stop of EMD : too small amplitude')
else
            disp('forced stop of EMD : too small amplitude')
end
break
end



% sifting loop
while ~stop_sift & nbit<MAXITERATIONS


if(nbit>MAXITERATIONS/5 & mod(nbit,floor(MAXITERATIONS/10))==0 & ~FIXE & nbit > 100)
            disp(['mode ',int2str(k),', iteration ',int2str(nbit)])
if exist('s')
                disp(['stop parameter mean value : ',num2str(s)])
end
            [im,iM] = extr(m);
            disp([int2str(sum(m(im) > 0)),' minima > 0; ',int2str(sum(m(iM) < 0)),' maxima < 0.'])
end


%sifting
        m = m - moyenne;


%computation of mean and stopping criterion
if FIXE
            [stop_sift,moyenne] = stop_sifting_fixe(t,m,INTERP);
elseif FIXE_H
            [stop_sift,moyenne,stop_count] = stop_sifting_fixe_h(t,m,INTERP,stop_count,FIXE_H);
else
            [stop_sift,moyenne,s] = stop_sifting(m,t,sd,sd2,tol,INTERP);
```

```
end

% display

if display_sifting
            [envminp,envmaxp,envmoyp] = envelope(t,mp,INTERP);
if FIXE |FIXE_H
                display_emd_fixe(t,m,mp,r,envminp,envmaxp,envmoyp,nbit,k,display_sifting)
else
                sxp=2*(abs(envmoyp))./(abs(envmaxp-envminp));
                sp = mean(sxp);

display_emd(t,m,mp,r,envminp,envmaxp,envmoyp,s,sp,sxp,sdt,sd2t,nbit,k,display_sifting,stop_sift)
end
end



        mp = m;
        nbit=nbit+1;
        NbIt=NbIt+1;


if(nbit==(MAXITERATIONS-1) & ~FIXE & nbit > 100)
if exist('s')
                warning(['forced stop of sifting : too many iterations... mode ',int2str(k),'. stop parameter
mean value : ',num2str(s)])
else
                warning(['forced stop of sifting : too many iterations... mode ',int2str(k),'.'])
end
end


end% sifting loop
    imf(k,:) = m;
if display_sifting
        disp(['mode ',int2str(k),' stored'])
end
    nbits(k) = nbit;
    k = k+1;



    r = r - m;
    nbit=0;



end%main loop

if sum(r.^2) & ~any(mask)
    imf(k,:) = r;
end

ort = io(x,imf);

if display_sifting
  close
end

%------------------------------------------------------------------------------------------------

function stop = stop_EMD(r)
    [indmin,indmax,indzer] = extr(r);
    ner = length(indmin) + length(indmax);
    stop = ner <3;



%------------------------------------------------------------------------------------------------



function [stop,envmoy,s]= stop_sifting(m,t,sd,sd2,tol,INTERP)
try
        [envmin,envmax,envmoy,indmin,indmax,indzer] = envelope(t,m,INTERP);
        nem = length(indmin) + length(indmax);
        nzm = length(indzer);

% evaluation of mean zero
        sx=2*(abs(envmoy))./(abs(envmax-envmin));
        s = mean(sx);
```

```
        stop = ~((mean(sx > sd) > tol | any(sx > sd2) | (abs(nzm-nem)>1)) & (nem > 2));
catch
        disp(lasterr)
        stop = 1;
        envmoy = zeros(1,length(m));
%       disp(['catch : ',lasterr])
        s = NaN;
end



%---------------------------------------------------------------------------------------------------
function [stop,moyenne]= stop_sifting_fixe(t,m,INTERP)
try
        [envmin,envmax,moyenne] = envelope(t,m,INTERP);
        stop = 0;
%       disp('try ok')
catch
        moyenne = zeros(1,length(m));
        stop = 1;
%       disp(['catch : ',lasterr])
end



%---------------------------------------------------------------------------------------------------
function [stop,moyenne,stop_count]= stop_sifting_fixe_h(t,m,INTERP,stop_count,FIXE_H)
try
        [envmin,envmax,moyenne,indmin,indmax,indzer] = envelope(t,m,INTERP);
        nem = length(indmin) + length(indmax);
        nzm = length(indzer);


if (abs(nzm-nem)>1)
            stop = 0;
            stop_count = 0;
else
            stop_count = stop_count+1;
            stop = (stop_count == FIXE_H);
end
%       disp('try ok')
catch
        moyenne = zeros(1,length(m));
        stop = 1;
%       disp(['catch : ',lasterr])
end



%---------------------------------------------------------------------------------------------------


function display_emd(t,m,mp,r,envmin,envmax,envmoy,s,sb,sx,sdt,sd2t,nbit,k,display_sifting,stop_sift)
    subplot(4,1,1)
    plot(t,mp);hold on;
    plot(t,envmax,'--k');plot(t,envmin,'--k');plot(t,envmoy,'r');
    title(['IMF ',int2str(k),';   iteration ',int2str(nbit),' before sifting']);
    set(gca,'XTick',[])
    hold  off
    subplot(4,1,2)
    plot(t,sx)
    hold on
    plot(t,sdt,'--r')
    plot(t,sd2t,':k')
    title('stop parameter')
    set(gca,'XTick',[])
    hold off
    subplot(4,1,3)
    plot(t,m)
    title(['IMF ',int2str(k),';   iteration ',int2str(nbit),' after sifting']);
    set(gca,'XTick',[])
    subplot(4,1,4);
    plot(t,r-m)
    title('residue');
    disp(['stop parameter mean value : ',num2str(sb),' before sifting and ',num2str(s),' after'])
if stop_sift
        disp('last iteration for this mode')
end
if display_sifting == 2
      pause(0.01)
else
      pause
end



%---------------------------------------------------------------------------------------------------

function display_emd_fixe(t,m,mp,r,envmin,envmax,envmoy,nbit,k,display_sifting)
```

```
    subplot(3,1,1)
    plot(t,mp);hold on;
    plot(t,envmax,'--k');plot(t,envmin,'--k');plot(t,envmoy,'r');
    title(['IMF ',int2str(k),';   iteration ',int2str(nbit),' before sifting']);
    set(gca,'XTick',[])
    hold  off
    subplot(3,1,2)
    plot(t,m)
    title(['IMF ',int2str(k),';   iteration ',int2str(nbit),' after sifting']);
    set(gca,'XTick',[])
    subplot(3,1,3);
    plot(t,r-m)
    title('residue');
if display_sifting == 2
      pause(0.01)
else
      pause
end

%-------------------------------------------------------------------------------------------------------


function [envmin, envmax,envmoy,indmin,indmax,indzer] = envelope(t,x,INTERP)
%computes envelopes and mean with various interpolations


NBSYM = 2;
DEF_INTERP = 'spline';



if nargin < 2
    x = t;
    t = 1:length(x);
    INTERP = DEF_INTERP;
end


if nargin == 2
if ischar(x)
        INTERP = x;
        x = t;
        t = 1:length(x);
end
end


if ~ischar(INTERP)
    error('interp parameter must be ''linear''', ''cubic'' or ''spline''')
end


if ~any(strcmpi(INTERP,{'linear','cubic','spline'}))
    error('interp parameter must be ''linear''', ''cubic'' or ''spline''')
end


if min([size(x),size(t)]) > 1
    error('x and t must be vectors')
end
s = size(x);
if s(1) > 1
    x = x';
end
s = size(t);
if s(1) > 1
    t = t';
end
if length(t) ~= length(x)
    error('x and t must have the same length')
end


lx = length(x);
[indmin,indmax,indzer] = extr(x,t);



%boundary conditions for interpolation


[tmin,tmax,xmin,xmax] = boundary_conditions(indmin,indmax,t,x,NBSYM);


% definition of envelopes from interpolation


envmax = interp1(tmax,xmax,t,INTERP);
envmin = interp1(tmin,xmin,t,INTERP);


if nargout > 2
```

```
    envmoy = (envmax + envmin)/2;
end



%------------------------------------------------------------------------------------------

function [tmin,tmax,xmin,xmax] = boundary_conditions(indmin,indmax,t,x,nbsym)
% computes the boundary conditions for interpolation (mainly mirror symmetry)



    lx = length(x);


if (length(indmin) + length(indmax) < 3)
        error('not enough extrema')
end


if indmax(1) < indmin(1)
if x(1) > x(indmin(1))
            lmax = fliplr(indmax(2:min(end,nbsym+1)));
            lmin = fliplr(indmin(1:min(end,nbsym)));
            lsym = indmax(1);
else
            lmax = fliplr(indmax(1:min(end,nbsym)));
            lmin = [fliplr(indmin(1:min(end,nbsym-1))),1];
            lsym = 1;
end
else


if x(1) < x(indmax(1))
            lmax = fliplr(indmax(1:min(end,nbsym)));
            lmin = fliplr(indmin(2:min(end,nbsym+1)));
            lsym = indmin(1);
else
            lmax = [fliplr(indmax(1:min(end,nbsym-1))),1];
            lmin = fliplr(indmin(1:min(end,nbsym)));
            lsym = 1;
end
end


if indmax(end) < indmin(end)
if x(end) < x(indmax(end))
            rmax = fliplr(indmax(max(end-nbsym+1,1):end));
            rmin = fliplr(indmin(max(end-nbsym,1):end-1));
            rsym = indmin(end);
else
            rmax = [lx,fliplr(indmax(max(end-nbsym+2,1):end))];
            rmin = fliplr(indmin(max(end-nbsym+1,1):end));
            rsym = lx;
end
else
if x(end) > x(indmin(end))
            rmax = fliplr(indmax(max(end-nbsym,1):end-1));
            rmin = fliplr(indmin(max(end-nbsym+1,1):end));
            rsym = indmax(end);
else
            rmax = fliplr(indmax(max(end-nbsym+1,1):end));
            rmin = [lx,fliplr(indmin(max(end-nbsym+2,1):end))];
            rsym = lx;
end
end


    tlmin = 2*t(lsym)-t(lmin);
    tlmax = 2*t(lsym)-t(lmax);
    trmin = 2*t(rsym)-t(rmin);
    trmax = 2*t(rsym)-t(rmax);


% in case symmetrized parts do not extend enough
if tlmin(1) > t(1) | tlmax(1) > t(1)
if lsym == indmax(1)
            lmax = fliplr(indmax(1:min(end,nbsym)));
else
            lmin = fliplr(indmin(1:min(end,nbsym)));
end
if lsym == 1
            error('bug')
end
        lsym = 1;
        tlmin = 2*t(lsym)-t(lmin);
        tlmax = 2*t(lsym)-t(lmax);
end


if trmin(end) < t(lx) | trmax(end) < t(lx)
```

```
if rsym == indmax(end)
            rmax = fliplr(indmax(max(end-nbsym+1,1):end));
else
            rmin = fliplr(indmin(max(end-nbsym+1,1):end));
end
if rsym == lx
        error('bug')
end
        rsym = lx;
        trmin = 2*t(rsym)-t(rmin);
        trmax = 2*t(rsym)-t(rmax);
end


    xlmax =x(lmax);
    xlmin =x(lmin);
    xrmax =x(rmax);
    xrmin =x(rmin);


    tmin = [tlmin t(indmin) trmin];
    tmax = [tlmax t(indmax) trmax];
    xmin = [xlmin x(indmin) xrmin];
    xmax = [xlmax x(indmax) xrmax];

%---------------------------------------------------------------------------------------------------


function [indmin, indmax, indzer] = extr(x,t);
%extracts the indices corresponding to extrema


if(nargin==1)
  t=1:length(x);
end


m = length(x);


if nargout > 2
    x1=x(1:m-1);
    x2=x(2:m);
    indzer = find(x1.*x2<0);


if any(x == 0)
      iz = find( x==0 );
      indz = [];
if any(diff(iz)==1)
        zer = x == 0;
        dz = diff([0 zer 0]);
        debz = find(dz == 1);
        finz = find(dz == -1)-1;
        indz = round((debz+finz)/2);
else
        indz = iz;
end
      indzer = sort([indzer indz]);
end
end


d = diff(x);


n = length(d);
d1 = d(1:n-1);
d2 = d(2:n);
indmin = find(d1.*d2<0 & d1<0)+1;
indmax = find(d1.*d2<0 & d1>0)+1;



% when two or more consecutive points have the same value we consider only one extremum in the middle of the
constant area


if any(d==0)


  imax = [];
  imin = [];


  bad = (d==0);
  dd = diff([0 bad 0]);
  debs = find(dd == 1);
  fins = find(dd == -1);
if debs(1) == 1
if length(debs) > 1
      debs = debs(2:end);
      fins = fins(2:end);
```

```
else
      debs = [];
      fins = [];
end
end
if length(debs) > 0
if fins(end) == m
if length(debs) > 1
        debs = debs(1:(end-1));
        fins = fins(1:(end-1));


else
        debs = [];
        fins = [];
end
end
end
  lc = length(debs);
if lc > 0
for k = 1:lc
if d(debs(k)-1) > 0
if d(fins(k)) < 0
          imax = [imax round((fins(k)+debs(k))/2)];
end
else
if d(fins(k)) > 0
          imin = [imin round((fins(k)+debs(k))/2)];
end
end
end
end


if length(imax) > 0
    indmax = sort([indmax imax]);
end


if length(imin) > 0
    indmin = sort([indmin imin]);
end


end


%-------------------------------------------------------------------------------------------------


function
[x,t,sd,sd2,tol,display_sifting,sdt,sd2t,ner,nzr,lx,r,imf,k,nbit,NbIt,MAXITERATIONS,FIXE,FIXE_H,MAXMODES,INTERP
,mask] = init(varargin)


x = varargin{1};
if nargin == 2
if strcmp(class(varargin{2}),'struct')
        inopts = varargin{2};
else
        error('when using 2 arguments the first one is the analysed signal x and the second one is a struct
object describing the options')
end
elseif nargin > 2
try
        inopts = struct(varargin{2:end});
catch
        error('bad argument syntax')
end
end


% default for stopping
defstop = [0.05,0.5,0.05];


opt_fields = {'t','stop','display','maxiterations','fix','maxmodes','interp','fix_h','mask'};


defopts.stop = defstop;
defopts.display = 0;
defopts.t = 1:max(size(x));
defopts.maxiterations = 2000;
defopts.fix = 0;
defopts.maxmodes = 0;
defopts.interp = 'spline';
defopts.fix_h = 0;
defopts.mask = 0;


opts = defopts;
```

```
if(nargin==1)
    inopts = defopts;
elseif nargin == 0
    error('not enough arguments')
end



names = fieldnames(inopts);
for nom = names'
if length(strmatch(char(nom), opt_fields)) == 0
       error(['bad option field name: ',char(nom)])
end
    eval(['opts.',char(nom),' = inopts.',char(nom),';'])
end


t = opts.t;
stop = opts.stop;
display_sifting = opts.display;
MAXITERATIONS = opts.maxiterations;
FIXE = opts.fix;
MAXMODES = opts.maxmodes;
INTERP = opts.interp;
FIXE_H = opts.fix_h;
mask = opts.mask;


S = size(x);
if ((S(1) > 1) & (S(2) > 1)) | (length(S) > 2)
  error('x must have only one row or one column')
end


if S(1) > 1
  x = x';
end


S = size(t);
if ((S(1) > 1) & (S(2) > 1)) | (length(S) > 2)
  error('option field t must have only one row or one column')
end


if S(1) > 1
  t = t';
end


if (length(t)~=length(x))
  error('x and option field t must have the same length')
end


S = size(stop);
if ((S(1) > 1) & (S(2) > 1)) | (S(1) > 3) | (S(2) > 3) | (length(S) > 2)
  error('option field stop must have only one row or one column of max three elements')
end


if ~all(isfinite(x))
    error('data elements must be finite')
end


if S(1) > 1
  stop = stop';
  S = size(stop);
end


if S(2) < 3
  stop(3)=defstop(3);
end


if S(2) < 2
  stop(2)=defstop(2);
end



if ~ischar(INTERP)
    error('interp field must be ''linear'', ''cubic'' or ''spline''')
end


if ~any(strcmpi(INTERP,{'linear','cubic','spline'}))
    error('interp field must be ''linear'', ''cubic'' or ''spline''')
```

```
end

%special procedure when a masking signal is specified
if any(mask)
    S = size(mask);
if min(S) > 1
        error('masking signal must have the same dimension as the analyzed signal x')
end
if S(1) > 1
        mask = mask';
end
if max(S) ~= max(size(x))
        error('masking signal must have the same dimension as the analyzed signal x')
end
    opts.mask = 0;
    imf1 = emd(x+mask,opts);
    imf2 = emd(x-mask,opts);
if size(imf1,1) ~= size(imf2,1)
        warning(['the two sets of IMFs have different sizes: ',int2str(size(imf1,1)),' and
',int2str(size(imf2,1)),' IMFs.'])
end
    S1 = size(imf1,1);
    S2 = size(imf2,1);
if S1 ~= S2
if S1 < S2
            tmp = imf1;
            imf1 = imf2;
            imf2 = tmp;
end
        imf2(max(S1,S2),1) = 0;
end
    imf = (imf1+imf2)/2;


end



sd = stop(1);
sd2 = stop(2);
tol = stop(3);


lx = length(x);


sdt = sd*ones(1,lx);
sd2t = sd2*ones(1,lx);


if FIXE
    MAXITERATIONS = FIXE;
if FIXE_H
        error('cannot use both ''fix'' and ''fix_h'' modes')
end
end


% number of extrema and zero-crossings in residual
ner = lx;
nzr = lx;


r = x;


if ~any(mask) % if a masking signal is specified "imf" already exists at this stage
    imf = [];
end
k = 1;


% iterations counter for extraction of 1 mode
nbit=0;


% total iterations counter
NbIt=0;
```